Escola de Camins
Escola Tècnica Superior d'Enginyeria de Camins, Canals i Ports
UPC BARCELONATECH

TREBALL FINAL DE GRAU

Implementació i comparació d'algoritmes per la obtenció del camí òptim. Aplicació en sistemes de weather routing

Treball realitzat per:
Lluís Martorell Pons

Dirigit per:
Manel Grifoll Colls i Francesc Robusté Antón

Grau en:
Enginyeria Civil

Barcelona, Juliol/2017

Departament d'Enginyeria Civil i Ambiental

TREBALL DE FI DE GRAU

IMPLEMENTACIÓ I COMPARACIÓ D'ALGORITMES PER LA OBTENCIÓ DEL CAMÍ ÒPTIM. APLICACIÓ EN SISTEMES DE WEATHER ROUTING

Autor de la memòria:

Martorell Pons, Lluís

supervisat per:

Prof. **Grifoll Colls, Manel**

Prof. **Robusté Antón, Franscesc**

Grau en Enginyeria Civil

Escola Tècnica Superior d'Enginyers de Camins, Canals i Ports de Barcelona

Universitat Politècnica de Catalunya

Juliol 2017

Abstracto

En el ámbito de la ingeniería, especialmente en Transporte y Logística, un problema muy común es el cálculo del camino mínimo por ejemplo en rutas de envío de mercancías, en desplazamientos entre ciudades, en transporte marítimo, etc. A lo largo de la historia se han desarrollado gran cantidad de algoritmos para resolver este problema y el más famosos es el conocido algoritmo Dijkstra.

Este trabajo tiene dos objetivos, uno centrado en el estudio y comparación de los algoritmos Dijkstra y A* y otro en el estudio del beneficio de la aplicación de un sistema de weather routing, llamado SIMROUTE, a trayectos de corta duración, basándose en la ruta: Barcelona – Palma de Mallorca.

Los resultados obtenidos muestran la utilidad de la función heurística en el algoritmo A* respecto al algoritmo Dijkstra y la reducción de los tiempos de computación al ser implementada en el programa SIMROUTE.

Además, una vez analizadas las simulaciones realizadas se ha determinado la mejora en 4 de cada 25 viajes realizados en años anteriores, si la utilización de sistemas de weather routing fuera implementada en los viajes realizados en la ruta Barcelona- Palma de Mallorca por las compañías navieras actuales.

Agradecimientos

Primero me gustaría dar las gracias a mis supervisores Manel Grifoll Colls y Francesc Robusté Antón por el apoyo y las ideas aportadas durante el desarrollo del proyecto. También a mis antiguos compañeros de piso, Marco, Miki y Tià, que me soportaron y con los que compartí grandísimos momentos, en especial a Marco Ojer que siempre me motivó en los momentos necesarios. No puedo dejar sin mencionar a mis compañeros de piso actuales, Elia, Laia y Maite, que siguen manteniendo el nivel del anterior piso. Y, por último, a Martí Montesinos, que compartió conmigo un maravilloso Erasmus en Dinamarca y no aceptaría no ser mencionado.

Índice

1. Introducción

Un problema muy común en el ámbito de la ingeniería, es el denominado, problema del camino mínimo. Este problema consiste en encontrar el camino entre 2 vértices de un grafo, de tal forma que el coste sea el mínimo posible, ver figura 2. El coste entre 2 vértices viene determinado por diferentes factores y es elegido por el diseñador del problema en cuestión. Por ejemplo, en el caso de un mapa, ver figura 1, los vértices representan las ciudades; las aristas, las carreteras, y el coste podría ser la distancia o bien los litros de combustible consumidos.

Figura 1: Red de carreteras de Navarra, considerando las ciudades y pueblos como nodos y las conexiones entre ellas como aristas se definiría el grafo de las carreteras en Navarra.

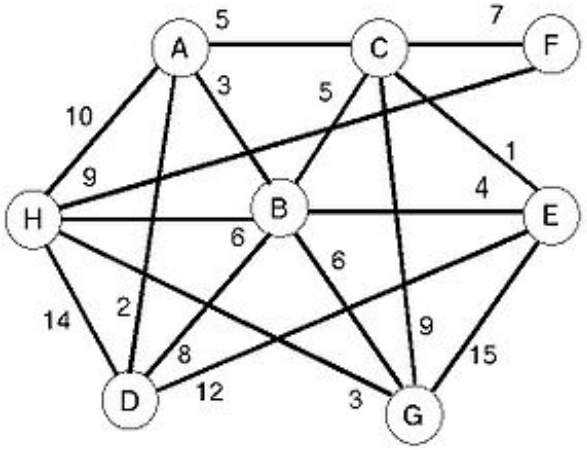


Figura 2: Ejemplo de grafo en el cual se quiere encontrar el camino mínimo entre dos de los nodos que se seleccionen.

Así pues, para plantear el problema se debe definir: el grafo, el cual incluye vértices, aristas y costes, así como el punto de origen y el punto final. La solución al problema de encontrar el camino óptimo entre los puntos origen y destino definidos es realizada mediante los denominados algoritmos de búsqueda del camino óptimo.

1.1. Algoritmos de camino óptimo

A lo largo de la historia se han desarrollado numerosos métodos, para resolver este problema. Los más conocidos son los algoritmos Dijkstra y A*, los cuales serán descritos más detalladamente en el capítulo 2, al centrarse este trabajo en estos dos algoritmos.

Otros algoritmos conocidos son, por ejemplo, el algoritmo Bellman-Ford, que es muy similar al Dijkstra con la diferencia que se pueden incluir pesos negativos en las aristas, para el caso de pesos positivos suele usarse el algoritmo Dijkstra, dado que el tiempo de computación es menor. También

encontramos el algoritmo Prim, que encuentra un árbol encubridor mínimo para un grafo, es decir el árbol en el que la suma de los costes de todas las aristas es mínima. También es conocido como algoritmo de Jarnik.

Otro enfoque son los algoritmos genéticos (Gúzman Luna, Arango Sánchez, & Jiménez Pinzón, 2012) basados en la evolución genética. A partir de una población inicial, en la que cada individuo representa un camino entre el punto inicial y el final, y mediante el uso de distintas operaciones como la selección de los mejores individuos; el cruzamiento entre ellos, es decir, la combinación de tramos de los individuos seleccionados; la mutación de ellos, que significa generar pequeñas variaciones y la generación de individuos aleatorios, se crean generaciones progresivamente hasta que encuentra una que cumpla ciertos criterios.

Otro método es el algoritmo de la colonia de hormigas, derivado de la inteligencia de enjambres. Este algoritmo se basa en el comportamiento de las hormigas al explorar el terreno en busca de una fuente de alimentos. Las hormigas recorren los caminos posibles dejando feromonas a su vuelta, estas feromonas atraen a otras hormigas al pasar por dicho camino, sin embargo, estas feromonas se evaporan con el paso del tiempo. Esto provoca que aumente la concentración de feromonas en los caminos más cortos, y en consecuencia las hormigas tienden al camino óptimo.

1.2. Algoritmos aplicados al cálculo de rutas optimas en barcos

Estos algoritmos pueden ser implementados en gran diversidad de sistemas para resolver diferentes problemas de la sociedad actual, como por ejemplo los sistemas de weather routing en el ámbito del tráfico marítimo para el cálculo de las rutas de los buques con tal de mejorar su rendimiento, reducir los costes de fuel, reducir el tiempo de viaje, etc. obteniendo con esto un beneficio económico a las empresas y también una reducción de las emisiones contaminantes, beneficioso para toda la sociedad.

El weather routing es la disciplina de producir la mejor ruta óptima con respecto al tiempo esperado de llegada, pasar por waypoints, la velocidad del navío, la potencia del motor usada, etc. para un viaje dado basado en la información de la predicción meteorológica para tormentas o tiempo inesperado y del funcionamiento del buque (Simonsen, Larsson, Mao, & Ringsberg, 31 mayo - 5 Junio, 2015).

En los sistemas de weather routing intervienen una gran cantidad de componentes y factores distintos, siendo estos sistemas considerablemente complejos. Con tal de entender mejor la estructura que una herramienta de weather routing tendría en la figura 3 se puede observar su desglose en diferentes categorías y subcomponentes que presentan una visión general.

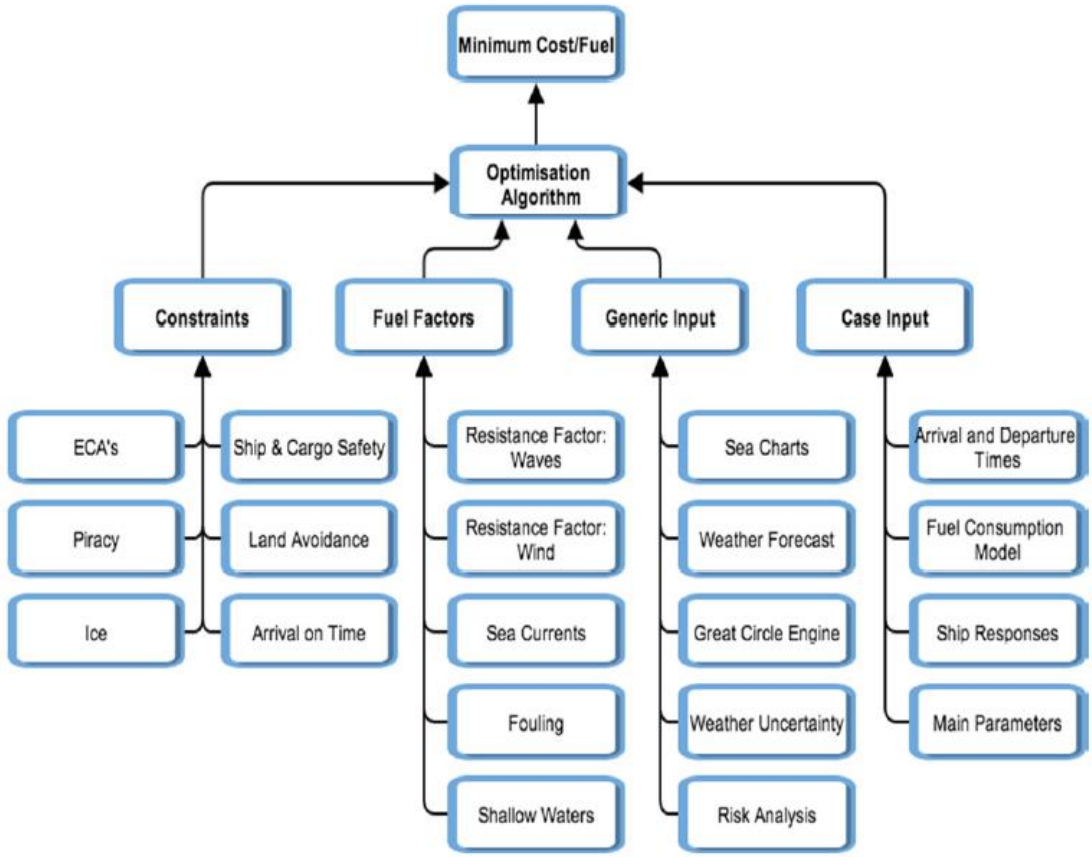


Figura 3: Vista general de la estructura de un sistema de weather routing. Fuente: **(Simonsen, Larsson, Mao, & Ringsberg, 31 mayo - 5 Junio, 2015)**

Vemos que para los sistemas de weather routing es necesaria la construcción de un método de optimización y éste depende mucho de la elección del algoritmo de optimización a usar, teniendo el algoritmo efecto en el tipo de soluciones, su calidad y el tiempo de computación necesario del sistema. En la comunidad marítima los algoritmos más utilizados son: el Algoritmo Dijkstra (Chu, Miller, & Hansen, 2015), la Programación Dinámica (Avgouleas, 2008) (Chen, Cardone, & Lacey, 1998) (Shao, Development of an intelligent tool for energy efficient and low environment impact shipping, 2013) y los Algoritmos Genéticos (Hu, Cai, Yang, & Shi, 2014) (Maki, y otros, 2011) (Marie & Courteille, 2099), además de los algoritmos tradicionalmente utilizados para la optimización del tráfico como Calculo de Variaciones (Bijlsma, 1975), el método Isopone (Hagiwara, 1982) y el método Isochrone (Klompstra, Olsde, & Van Brunschot, 1992). En Shao (Shao, Development of an intelligent tool for energy efficient and low environment impact shipping, 2013) se puede encontrar una breve descripción de todos estos algoritmos. Además, se siguen creando nuevos algoritmos y sistemas a través de las diferentes investigaciones en el campo de weather routing, como por ejemplo el algoritmo DIRECT (Dividing rectangles) (Larsson & Simonsen, 2014).

1.3. Descripción del documento

A continuación, se encuentra una descripción detallada del presente documento en el que se desglosan los temas tratados en las diferentes secciones venideras del trabajo, así como que material se encuentra en ellas.

En el capítulo 2, Métodos, se describen los algoritmos usados y estudiados en este trabajo, el Dijkstra y A*. Se encontrará una explicación de los algoritmos y sus orígenes, así como el pseudocódigo de ambos. Además, hay una explicación del programa utilizado en la simulación de rutas marítimas, el SIMROUTE, detallando los elementos que intervienen en su utilización, así como la base de su funcionamiento. Finalmente, Se encuentra una explicación del estudio estadístico que se pretende realizar en el presente trabajo con los datos obtenidos en las simulaciones de rutas de corta duración.

En el capítulo 3, Resultados, se encuentra una serie de gráficas y tablas que resumen todos los datos obtenidos de comparar los algoritmos Dijkstra y A* así como de las simulaciones llevadas a cabo con el SIMROUTE. Para los algoritmos Dijkstra y A* se han recogido 3 datos: los nodos abiertos (NA), los nodos cerrados (NC) y el tiempo de computación para: variaciones lineales de los elementos presentes en el mallado, posición de los nodos origen y destino o tamaño de un obstáculo, y para 6 geometrías de creciente complejidad. Para el SIMROUTE se han recogido 2 variables, el tiempo de viaje y las millas recorridas, para 3 casos diferentes: la ruta de distancia mínima sin considerar el efecto del oleaje, la ruta de distancia mínima teniendo en cuenta los efectos del oleaje y la ruta optimizada. Esta se ha realizado para dos escenarios: una ruta de corta duración, menos de 1 día de duración del viaje, entre los puertos de Barcelona –Palma de Mallorca y para una ruta de larga duración, 3 días de viaje o más, entre los puertos de Peñíscola (España) – Chania (Grecia).

En el capítulo 4, Discusión y conclusiones, se reflexiona sobre los datos obtenidos en Resultados tanto para la aplicación teórica de los algoritmos Dijkstra y A* como su aplicación en simulación de rutas marítimas o de los beneficios de aplicar sistemas de optimización de rutas en viajes de barco. También se discuten las limitaciones de este trabajo, así como las mejoras realizables en un futuro.

2. Métodos

2.1. Descripción algoritmo Dijkstra

El algoritmo Dijkstra, también llamado algoritmo de caminos mínimos, fue concebido en 1956 por el científico de la computación Edsger Wybe Dijkstra y publicado en 1959 (Dijkstra, 1959). Se trata de un algoritmo de búsqueda de grafos que resuelve el single-source shortest path problem, es decir, se trata de un algoritmo para la determinación, dado un grafo con pesos en cada arista, del camino más corto desde un vértice origen al resto.

Hay que tener en cuenta una serie de consideraciones a la hora de aplicar el algoritmo Dijkstra:
- Todos los pesos de las aristas deben ser positivos.
- Es necesario que el grafo sea conexo, es decir, para cualquier par de vértices o nodos u y v del grafo debe existir al menos una sucesión de nodos adyacentes, que no se repitan nodos, del nodo u al v.
- Sirve tanto para grafos dirigidos como no dirigidos.
- Es un algoritmo voraz que genera uno a uno los caminos de un nodo origen al resto en orden creciente de longitud. Por tanto, si queremos utilizarlo para determinar el camino mínimo entre un nodo origen y un nodo final hay que pararlo una vez el camino mínimo al nodo destino ha sido determinado.

Antes de profundizar en el pseudocódigo del algoritmo Dijkstra es importante saber cómo funciona este. El algoritmo Dijkstra se basa en el principio de optimalidad de Bellman que dicta que "dada una secuencia óptima de decisiones, toda sub-secuencia de ella es, a su vez, óptima". Lo que el algoritmo Dijkstra hace en cada iteración es calcular el próximo nodo que se encuentra a menor distancia del conjunto de nodos ya analizados, ponerlo como visitado y actualizar los costes del resto de nodos. Y va iterando hasta tener todos los nodos como visitados.

A continuación, se encuentra el pseudocódigo del algoritmo Dijkstra:

```
1.  método Dijkstra (Grafo, origen, destino):
2.      dist[origen] = 0
3.      creamos cola Cerrados, NC = ∅
4.      creamos cola Abiertos, NA = ∅
5.      añadimos origen a la cola NA
6.      mientras F ≠ ∅ :
7.          sacamos el elemento u de la cola NA tal que dist[u] sea mínima
8.          añadimos u a la cola NC
9.          para cada vértice v adyacente a u en el Grafo:
10.             sea w el peso entre vértice (u, v)
11.             si v ∉ NC:
12.                 Relajación (u, v, w)
```

```
1. método Relajación (actual, adyacente, peso):
2.     si distancia [actual] + peso < distancia [adyacente]
3.         distancia [adyacente] = distancia [actual] + peso
4.         añadimos adyacente a la cola NA
```

2.2. Descripción algoritmo A*

Propuesto por Peter Hart, Nils Nilsson y Bertram Raphael del Standford Research Institute (SRI International) en 1968. Fue creado en un intento de mejorar al robot del SRI International, Shakey the Robot, el cuál debía navegar a través de una habitación con obstáculos. Se trata de un algoritmo de búsqueda informada en grafos el cuál, empezando desde el nodo origen, va construyendo un árbol de caminos, expandiendo uno de los caminos en cada iteración del algoritmo hasta llegar con uno de los caminos al nodo destino. Se trata de una mejora del algoritmo Dijkstra mediante la implementación de una función heurística, $h(n)$ dentro de la función de costes, $f(n)$. Así la función del coste total es calculada como:

$$f(n) = g(n) + h(n) \tag{1}$$

Donde $g(n)$ es el coste de la trayectoria desde el nodo inicial al nodo n y h(n) es la función heurística que estima el costo del mejor camino al nodo destino. Si $h(n) = 0$ se trata del algoritmo Dijkstra.

Hay que tener en cuenta una serie de consideraciones a la hora de aplicar el algoritmo A*:

- Todos los pesos de las aristas deben ser positivos.
- Es necesario que el grafo sea conexo, es decir, para cualquier par de vértices o nodos u y v del grafo debe existir al menos una sucesión de nodos adyacentes, que no se repitan nodos, del nodo u al v.
- Sirve tanto para grafos dirigidos como no dirigidos.
- Debe emplear una heurística admisible, es decir, que no sobreestime la distancia entre el nodo actual y el nodo destino.

A continuación, se encuentra el pseudocódigo del algoritmo A*:

```
1. método A* (Grafo, origen, destino):
2.     g(origen) = 0
3.     h(origen) = estimación heurística de la distancia a destino
6.     creamos cola Cerrados, NC, vacía
7.     creamos cola Abiertos, NA, vacía
8.     añadimos origen a la cola NA
9.     mientras NA no este vacío:
10.        sacamos el elemento de la cola NA tal que f[u] sea mínima
11.        añadimos u a la cola NC
12         generar cada nodo sucesor v del nodo u
13.        para cada sucesor v de u:
14.            establecer u como el padre de v
15.            h(v) = la estimación heurística de la distancia a destino
16.            g(v) = g(u) + w (peso)
```

17. f(v) = g(v) + h(v)
19. remover las ocurrencias de la lista cerrados
20. añadir v a la cola NA

2.3. Descripción SIMROUTE

El programa utilizado para la comparación de los algoritmos Dijkstra y A* en la optimización de rutas marítimas, así como para la realización de las simulaciones de rutas de corta y larga duración en este trabajo es llamado SIMROUTE y ha sido desarrollado por la UPC en lenguaje Matlab. Éste permite trabajar en todo el mediterráneo o en una zona de este dándole las longitudes y latitudes de la sección deseada. Es necesario introducirle los datos del oleaje de los días de los que se desee realizar una simulación entre dos puntos seleccionados. También se ha de definir la velocidad a la que circulará el navío.

Como introducción al programa en la figura 4 se encuentra un diagrama de flujo de éste en el que se puede observar el orden que sigue el programa durante su ejecución, así como el nombre de los scripts o funciones que realizan las diferentes acciones.

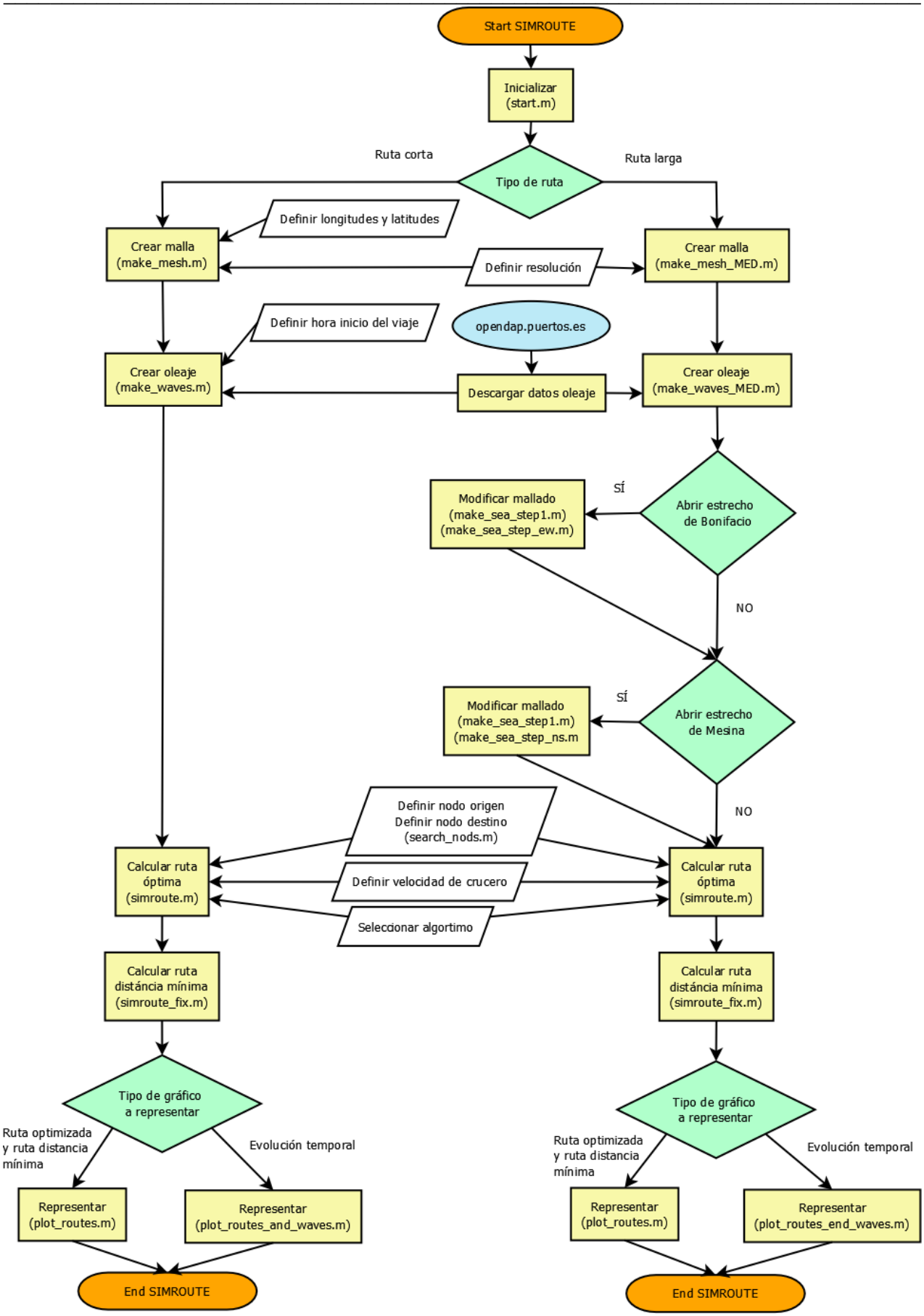


Figura 4: Diagrama de flujo del SIMROUTE

2.3.1. Descripción del algoritmo y discretización de la malla

Los algoritmos de búsqueda de camino usados en el programa son los mencionados Dijkstra y A*. Estos algoritmos son aplicados a un mallado donde cada nodo está conectado a un conjunto de puntos cercanos. A cada conexión entre nodos se asigna un peso relacionado con la distancia. La distancia ortodrómica se utiliza para las coordenadas esféricas de los nodos de la malla. Los algoritmos Dijkstra y A* en mallados rectangulares selecciona el vértice a visitar con el coste más bajo, calcula el coste a través de cada vecino no visitado, y actualiza los costes vecinos si son menores. El algoritmo Dijkstra ya ha sido utilizado previamente para cálculo de rutas de buques, por ejemplo, en (Mannarini, Coppini, Oddo, & Pinardi, 2013), (Montes, 2005).

Las posibles conexiones nodales por nodo pueden variar en función de la resolución de la malla. En consecuencia, la secuencia de aristas seguido por el camino más corto estará limitada por la resolución del mallado y los nodos conectados. La figura 5 muestra las aristas de conexión entre nodos mostradas por flechas para 4 esquemas diferentes: 4, 8, 16 y 24 aristas. Cada flecha representa una dirección potencial del navío a seguir. Se han realizado estudios sobre las diferentes resoluciones de mallado obteniendo conclusiones similares a (Mannarini, Coppini, Oddo, & Pinardi, 2013), donde se afirma que un mínimo de 16 aristas es necesario para trabajar en prototipos. (Grifoll M. , 2016)

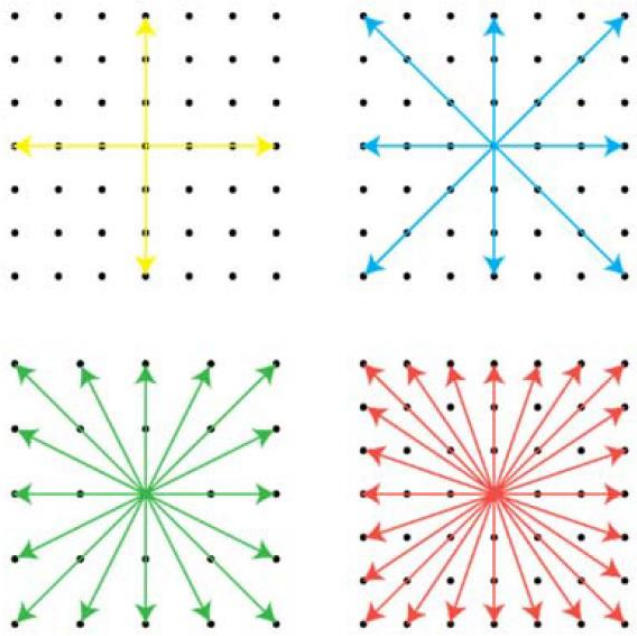


Figura 5: Esquema de la resolución de la malla en función del número de aristas por nodo. En amarillo 4 aristas por nodo, en azul 8 aristas por nodo, en verde 16 aristas por nodo y en rojo 24 bordes por nodo. Fuente: Manel Grifoll

2.3.2. Descripción del modelo de oleaje y efectos en la navegación

Recientemente, la implementación de avanzados modelos numéricos de oleaje proporciona parámetros de ola de alta resolución debido al incremento de la resolución del wind field (Cavaleri, 2007). En el programa SIMROUTE, es implementado el modelo de oleaje SWAN en el noroeste del mar Mediterráneo para generar outputs de olas modeladas que es usado como tormenta característica (Grifoll, y otros, 2016). El modelo de oleaje SWAN resuelve la ecuación del equilibrio de la acción de las olas simulando la generación de viento y propagación en aguas costeras y profundas. La figura 6 muestra la altura de ola significativa y la dirección del oleaje para un sistema

típico de baja presión localizado en noreste del mar mediterráneo occidental. Esta configuración sinóptica puede generar fuertes vientos y grandes alturas de ola significativa en el mar de Cataluña/Baleares. (Grifoll M. , 2016)

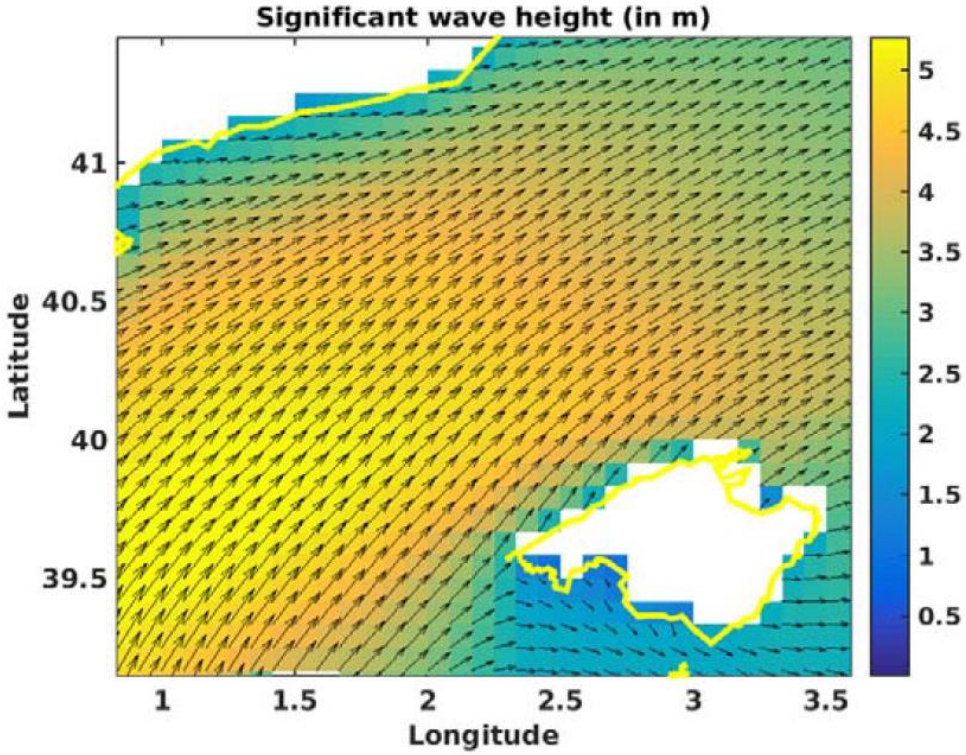


Figura 6: Altura de ola significativa y dirección del oleaje modelados en el mar catalán/balear. Las flechas muestran la dirección de propagación de las olas. Fuente: Manel Grifoll

La acción de las olas es el principal factor que afecta el rendimiento del buque (Hu, Cai, Yang, & Shi, 2014). El oleaje afecta a la movilidad del navío haciendo que disminuya el empuje de la hélice y añadiendo una resistencia en comparación con la ausencia de olas. (Grifoll M. , 2016) Una fórmula simple para incluir la reducción de la velocidad de la nave es sugerida por (Bowditch, 2002). La velocidad final es calculada en función de la velocidad no afectada más la reducción en función de los parámetros del oleaje:

$$v = v_0 - f(\Theta) \cdot H^2 \tag{2}$$

Donde H es la altura significante de ola i f es un parámetro en función de la dirección relativa del barco respecto a las olas. En la tabla 1 hay un listado de los valores del coeficiente f para las diferentes direcciones relativas del navío respeto a las olas.

Tabla 1: Valores del coeficiente f.

Ship-wave relative direction	Wave direction name from the ship point of view	f (in kn/ft^2)
$0^o \leq \Theta \leq 45^o$	Following seas	0.0083
$45^o < \Theta < 135^o$	Beam seas	0.0165
$135^o \leq \Theta \leq 225^o$	Head seas	0.0248
$225^o < \Theta < 270$	Beam seas	0.0165
$270^o \leq \Theta \leq 360^o$	Following seas	0.083

Por tanto, los parámetros del oleaje que intervienen en el SIMROUTE son dos: la altura significativa de ola y la dirección relativa del navío con el oleaje. En cuanto a las variables que son estudiadas una vez aplicado el programa son el tiempo de trayecto y las millas recorridas.

2.3.3. Obtención de los datos del oleaje

Los datos del oleaje utilizados son los proporcionados por el Ministerio de Fomento del Gobierno de España a través del catálogo que se encuentra en la página web opendap.puertos.es. Solo se dispone de datos desde el mes de junio de 2016 en adelante.

Los datos interesantes para este trabajo son los pertenecientes a tormentas relativamente grandes acontecidas en la ruta a estudiar. La fácil localización de las tormentas es posible gracias a la herramienta de la web de Puertos del Estado de Predicción de oleaje, nivel de mar; Boyas y mareógrafos, *www.puertos-es/es-es/oceanografia/Paginas/portus.aspx* en donde es posible visualizar los datos históricos del oleaje para un punto seleccionado del mapa. Generalmente las tormentas de distribuyen durante los primeros y los últimos meses del año, es decir, principalmente durante el invierno.

Para el correcto funcionamiento del SIMROUTE, para horas de inicio del viaje superiores a las 16:00h es necesario la introducción de los datos del siguiente día con tal de disponer de suficientes horas de oleaje para la simulación de la ruta corta.

2.4. Descripción estudio estadístico sobre impactos anuales

Una vez obtenidos los resultados de las simulaciones para las rutas de corta duración, este trabajo se propone realizar un sencillo estudio estadístico de los últimos años con el fin de calcular el número de viajes durante esos años que se habrían beneficiado de la aplicación de sistemas de optimización de rutas marítimas.

Con tal fin se busca definir una altura de ola significativa a partir de la cual la aplicación del programa implique un beneficio relevante y luego ver cuántas veces fue superada esta cota durante los años 2013, 2014, 2015, 2016 y la mitad del 2017. Para ello se utilizará la herramienta de la web de Puertos del Estado de Predicción de oleaje, nivel de mar; Boyas y mareógrafos en *www.puertos-es/es-es/oceanografia/Paginas/portus.aspx*, dedicada a los datos históricos, con la cual se puede acceder a los datos referentes a diferentes años de boyas ubicadas en el mar y también de puntos ficticios del mar de los cuales se han calculado los datos. En la figura 7 puede observarse esta distribución de boyas reales y puntos ficticios de los cuáles se disponen datos.

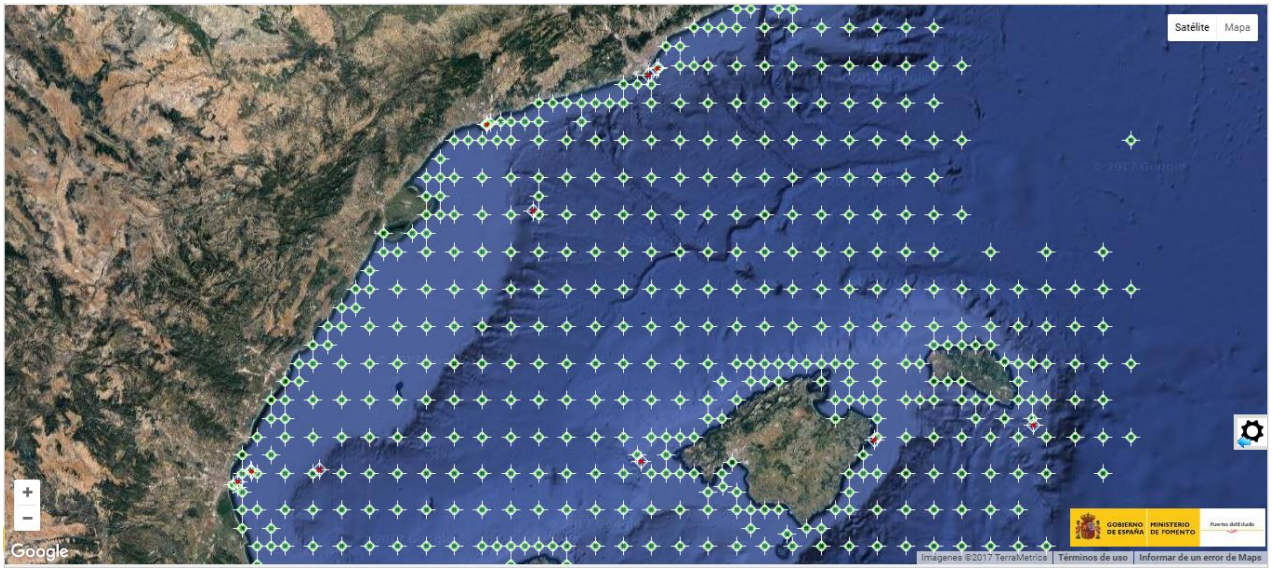


Figura 7: Puntos donde hay registro temporal (real o calculado) entre la costa de Mallorca y la de Barcelona. Fuente: Puertos del Estado

En la figura 8 se observa una aproximación del trazado de la ruta fija o de distancia mínima, en naranja, adaptada a las boyas más próximas a la ruta real. Además, se han señalado 3 boyas las cuales serán los puntos de los que se extraerán los datos, se analizarán y luego se contabilizaran todos los días en que al menos en una de las 3 boyas se superó la cota definida. La boya 1 corresponde al punto SIMAR (2110132), la boya 2 corresponde al punto SIMAR (2110126) y la boya 3 corresponde al punto SIMAR (2110118).

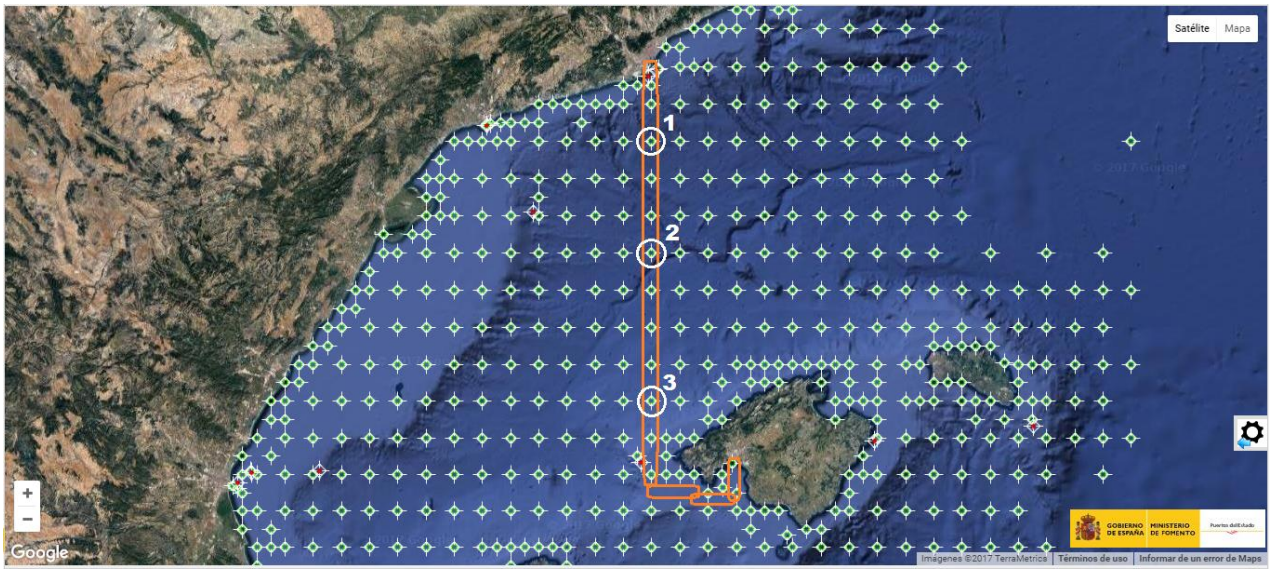


Figura 8: Visualización de la ruta de menor distancia y las boyas a estudiar. Fuente: Puertos del Estado y del autor

En la figura 9 se pueden observar los datos que proporciona la web de Puertos del Estado correspondientes al punto SIMAR (2110118) para el año 2016 si es seleccionada la casilla de *"datos horarios"* en la herramienta web.

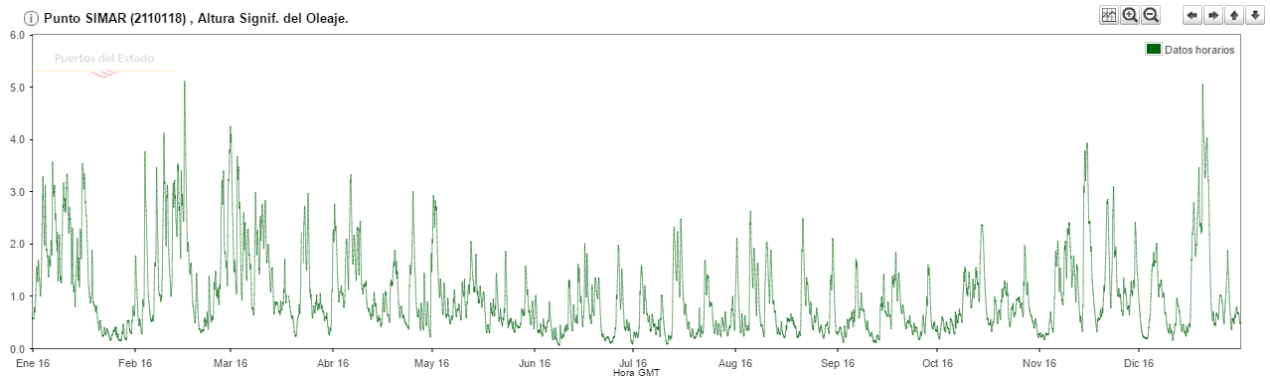


Figura 9: Datos horarios para el transcurso del año 2016 del punto SIMAR (2110118).
Fuente: Puertos del Estado y del autor.

Además, es necesario contabilizar el número de viajes realizados al año entre las dos rutas. Para simplificar el problema se ha considerado el punto de vista de una empresa, teniendo en cuenta así solo los viajes realizados por sus barcos, como ejemplo se ha considerado la compañía Trasmediterranea. Esta realiza actualmente: entre semana, 2 viajes de ida y 2 de vuelta al día entre Barcelona y Palma de Mallorca y los fines de semana, 1 viaje de ida y uno de vuelta por día. Se considerará que durante los pasados años se seguía este mismo sistema para realizar la estimación de viajes por año. En la tabla 2 se observan los viajes realizados por año desde el 2013 al 2017 (el 2017 solo se considera la mitad del año).

Tabla 2: Estimación del número de viajes realizados en el trayecto Barcelona – Palma de Mallorca.

Año	**2013**	**2014**	**2015**	**2016**	**2017**
Viajes estimados	1252	1252	1252	1254	622

3. Resultados

Los algoritmos y la metodología presentada en el apartado anterior han sido aplicados a casos puramente teóricos y algunos escenarios realistas de rutas marinas.

3.1. Comparación algoritmos Dijkstra y A* para casos sencillos

En el caso de este trabajo, para el algoritmo Dijkstra se estará trabajando con una malla rectangular, donde todos los costes son positivos; de coste 1 para movimientos horizontales o verticales y 1,4 para movimientos en diagonal; y el grafo será no dirigido. El programa Matlab es utilizado para realizar las simulaciones referentes a los algoritmos Dijkstra y A*, por lo que habrá que adaptar el pseudocódigo del algoritmo al lenguaje empleado en Matlab. El apartado 6.1 de los anexos contiene el código en Matlab elaborado por el autor de este trabajo para realizar las simulaciones.

Para el algoritmo A* se estará trabajando con una malla rectangular, donde todos los costes son positivos; la función de costes f(n) estará compuesta por la función g(n) que tendrá el coste hasta ese momento para llegar al nodo actual, considerando que el coste del movimiento horizontal es de 1 unidad y para movimientos en diagonal es de 1,4 unidades; la función h(n), la función heurística contará la cantidad de unidades horizontales y verticales que son necesarias para llegar al nodo destino desde el nodo actual y calculará el mínimo de movimientos horizontales y diagonales necesarios, otorgándoles el mismo coste en unidades que los que utiliza la función g(n); y el grafo será no dirigido. El programa Matlab es utilizado para realizar las simulaciones referentes a los algoritmos Dijkstra y A*, por lo que habrá que adaptar el pseudocódigo del algoritmo al lenguaje empleado en Matlab. El apartado 6.2 de los anexos contiene el código en Matlab elaborado para realizar las simulaciones.

Ambos algoritmos han sido aplicados a un conjunto de geometrías diferentes de complejidad creciente.

3.1.1. Geometría simple sin obstáculos

El grafo utilizado será una malla rectangular sin obstáculos de 20x20 (nodos) donde la posición del nodo origen es [10,8] y la del nodo destino [10,12]. En la figura 10 puede observarse la geometría descrita.

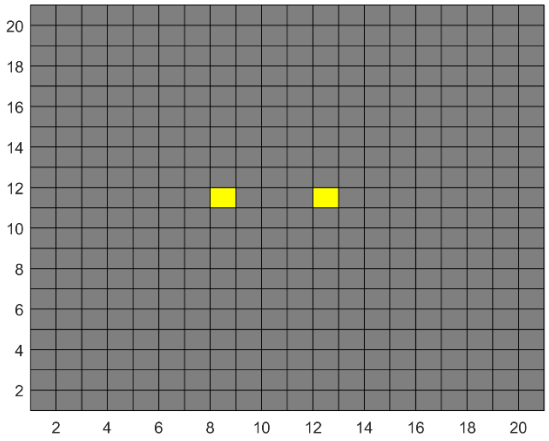


Figura 10: Malla para la geometría simple sin obstáculo.

En las figuras 11 y 12 se encuentran los resultados una vez aplicados los algoritmos Dijkstra y A*, respectivamente, al mallado anterior.

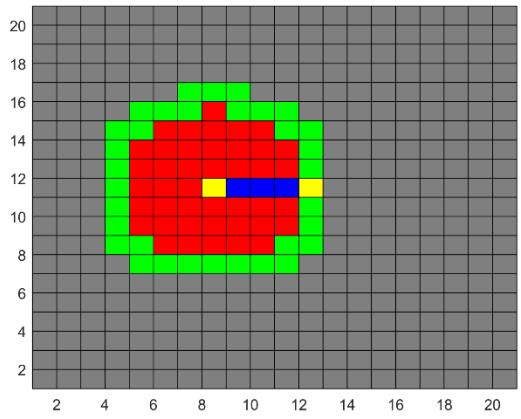


Figura 11: Camino óptimo encontrado aplicando Dijkstra a la geometría sin obstáculos.

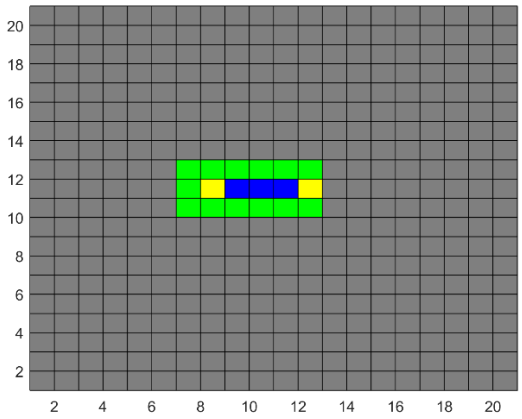


Figura 12: Camino óptimo encontrado aplicando A* a la geometría sin obstáculos..

3.1.2. Geometría simple con obstáculo vertical

El grafo utilizado será una malla rectangular de 20x20 (nodos) donde la posición del nodo origen es [10,8] y la del nodo destino [10,12] y un obstáculo vertical ubicado en la columna 10 con un ancho de 3 unidades, una arriba y una debajo del punto [10,10]. En la figura 13 se puede observar la geometría descrita.

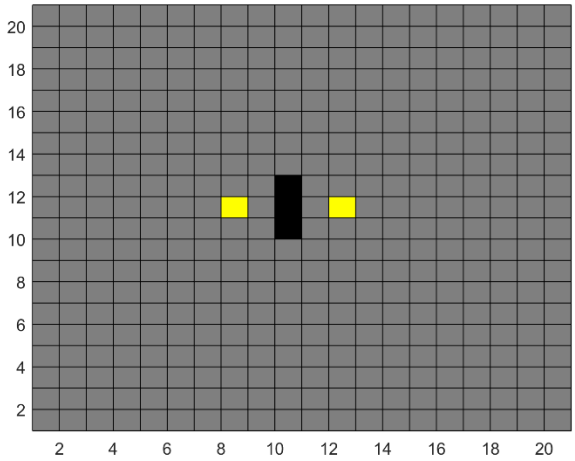


Figura 13: Malla para la geometría simple con obstáculo vertical.

En las figuras 14 y 15 se encuentran los resultados una vez aplicados los algoritmos Dijkstra y A*, respectivamente, al mallado anterior.

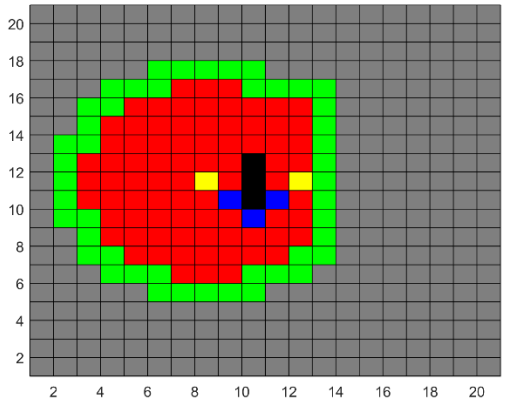


Figura 14: Camino óptimo encontrado aplicando Dijkstra a la geometría simple con obstáculo vertical.

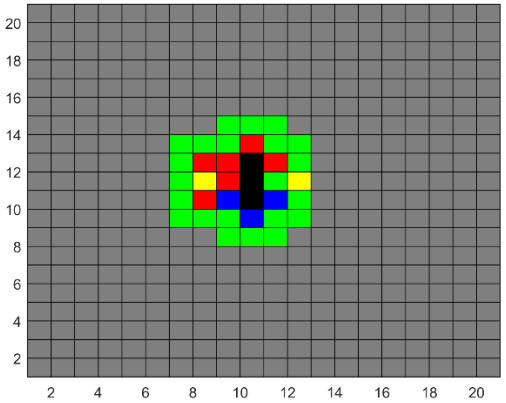
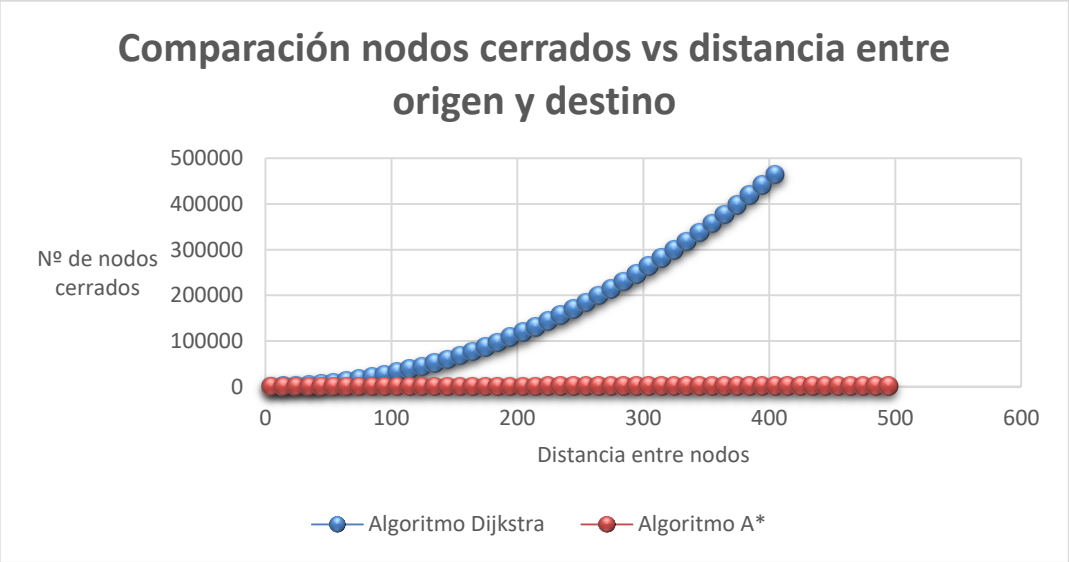


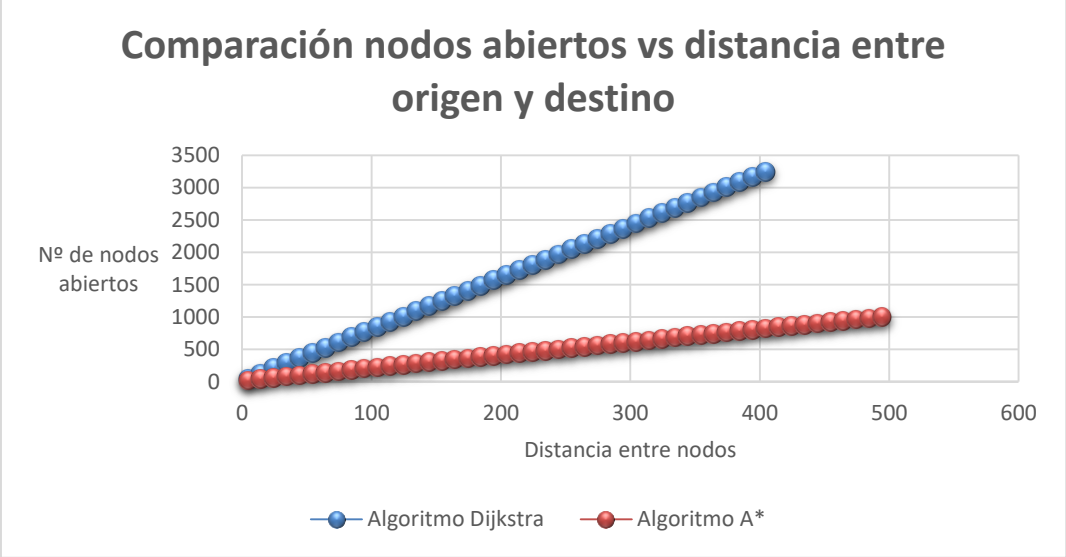
Figura 15: Camino óptimo encontrado aplicando A* a la geometría simple con obstáculo vertical.

3.1.3. Distanciamiento lineal entre nodo origen y nodo destino

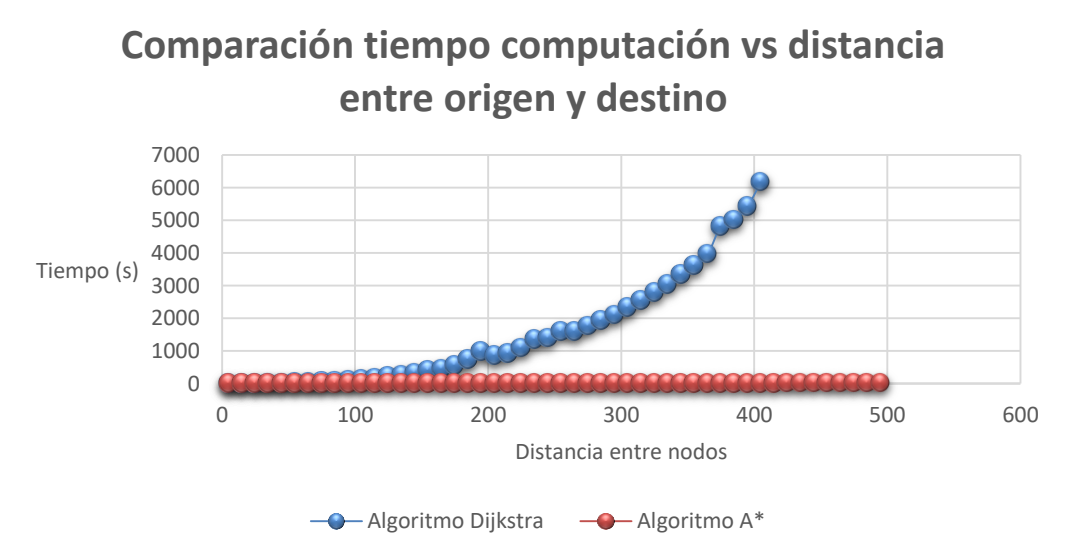
El grafo utilizado será una malla rectangular sin obstáculos de 1000x1000 (nodos) donde la posición del nodo origen es [500,500] y el nodo destino se irá distanciando mediante un incremento de 10 unidades, empezando en la posición [500,505], hasta llegar a la posición [500,995]. En la figura 16 se han graficado el número de nodos cerrados, el número de nodos abiertos y el tiempo de computación empleado en las iteraciones. La recolección de datos para el algoritmo Dijkstra han sido detenidos en la posición [500,905] del nodo de destino, debido al largo tiempo de computación necesario y al considerarse que los datos anteriores ya eran suficientes para extraer conclusiones.



a) Nº de nodos cerrados en función de la separación entre origen y destino



b) Nº de nodos abiertos en función de la separación entre origen y destino

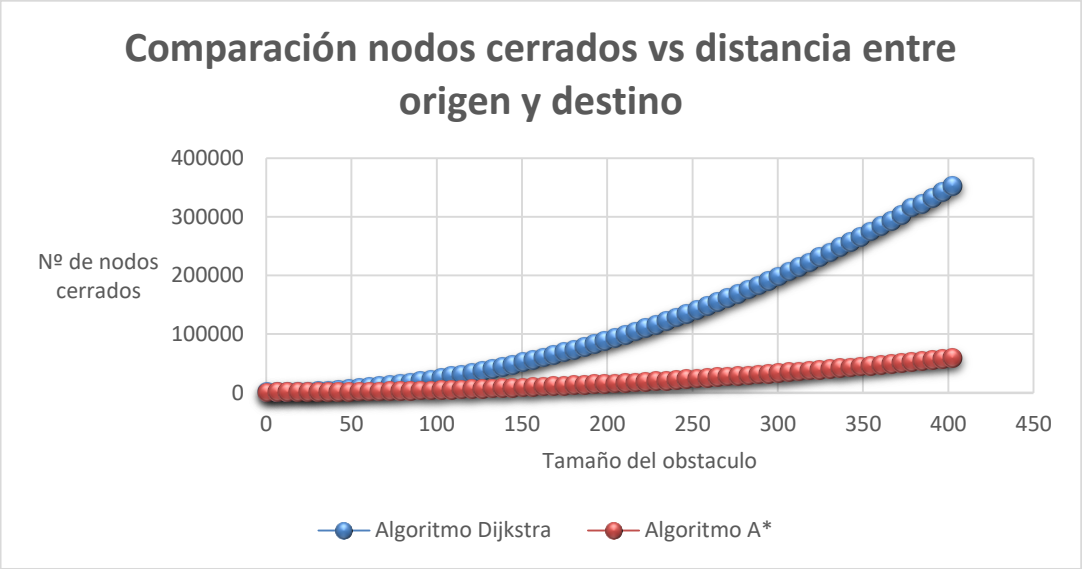


c)) Tiempo de computación en función de la separación entre origen y destino

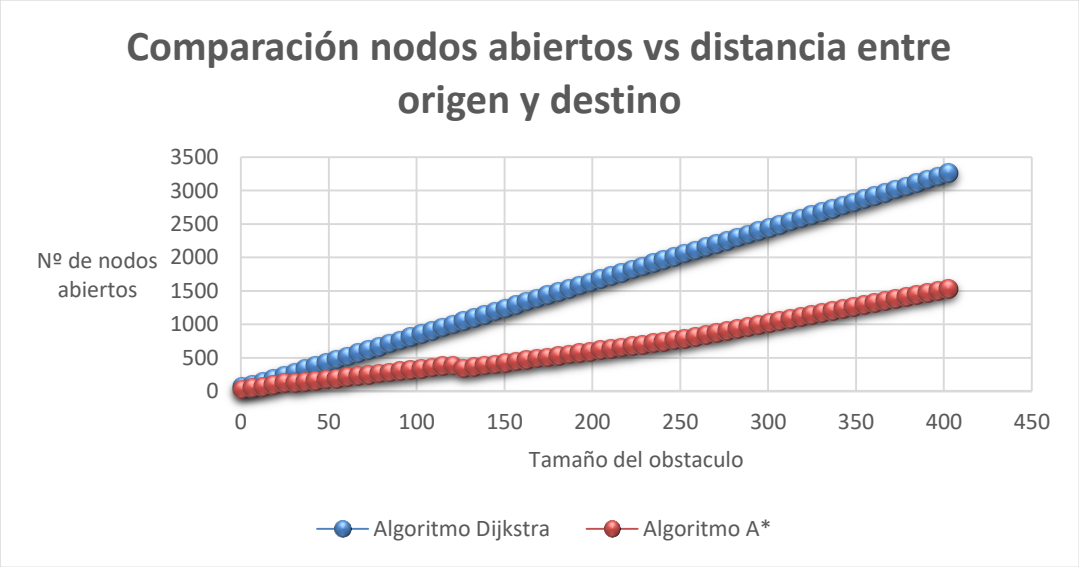
Figura 16: Resultados de aplicar Dijkstra y A* al distanciamiento lineal entre nodo origen y nodo destino

3.1.4. Aumento lineal del tamaño del obstáculo vertical

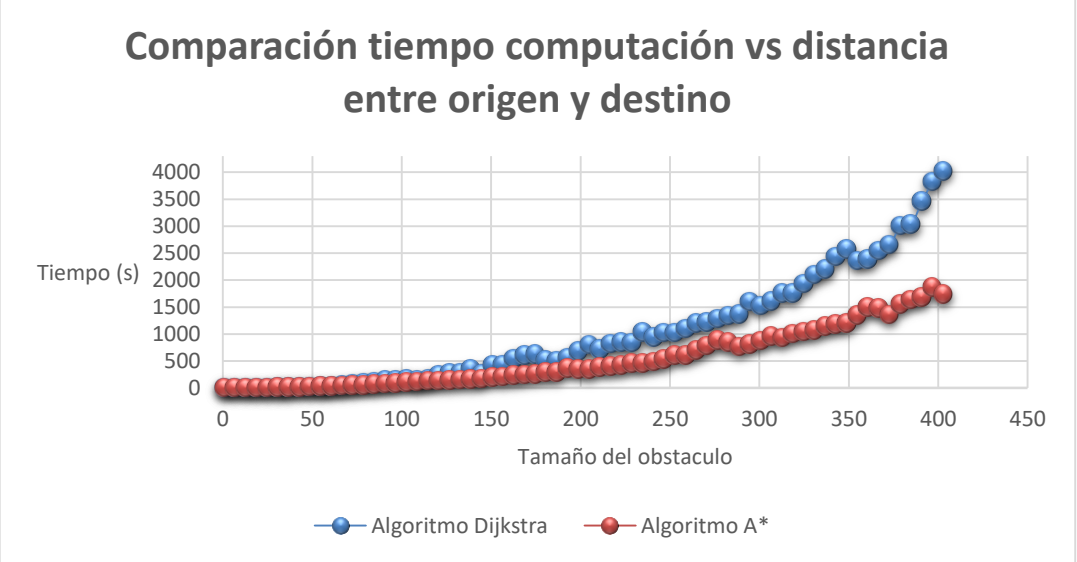
El grafo utilizado será una malla rectangular de 1000x1000 (nodos) donde la posición del nodo origen es [500,497] y la del nodo destino es [500,503] y un obstáculo vertical ubicado en la columna 500, con un tamaño inicial de un solo nodo, la posición [500,500], sufriendo un aumento de 6 unidades (3 por encima y 3 por debajo del nodo [500,500]) en cada iteración, hasta que el obstáculo alcance una amplitud de 403 unidades. En la figura 17 se han graficado el número de nodos cerrados, el número de nodos abiertos y el tiempo de computación empleado en las iteraciones.



a) Nº de nodos cerrados en función del tamaño del obstáculo vertical



b) Nº de nodos abiertos en función del tamaño del obstáculo vertical



c) Tiempo de computación en función del tamaño del obstáculo vertical

Figura 17: Resultados de aplicar Dijkstra y A* al aumento lineal del tamaño del obstáculo vertical.

3.1.5. Primera geometría: Nodo origen en el centro de la malla

El grafo utilizado será una malla rectangular sin obstáculos de 34x34 (nodos) donde la posición del nodo origen es [16,18] y la del nodo destino [16,29], habiendo una separación de 11 unidades entre ambos. En la figura 18 puede observarse la geometría descrita.

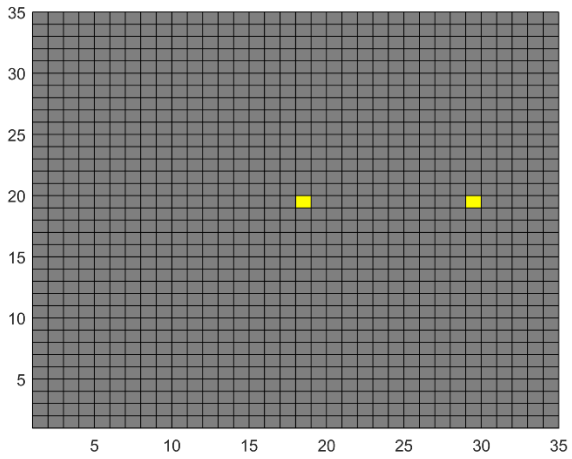


Figura 18: Malla para la primera geometría.

En las figuras 19 y 20 se encuentran los resultados una vez aplicados los algoritmos Dijkstra y A*, respectivamente, al mallado anterior.

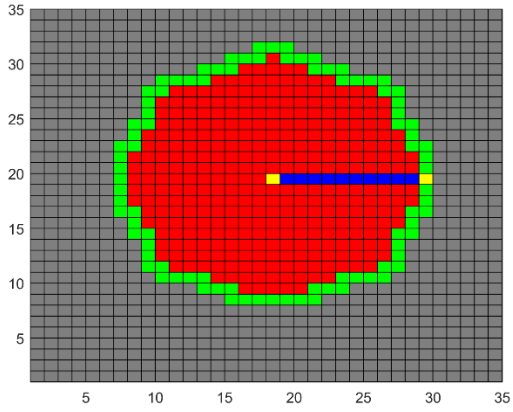


Figura 19: Camino óptimo encontrado aplicando Dijkstra a la primera geometría.

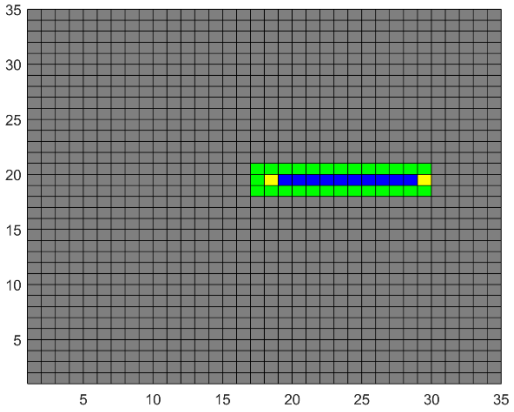


Figura 20: Camino óptimo encontrado aplicando A* a la primera geometría.

3.1.6. Segunda geometría: Nodo origen en el borde izquierdo de la malla

El grafo utilizado será una malla rectangular sin obstáculos de 34x34 (nodos) donde la posición del nodo origen es [16,1] y la del nodo destino [16,12], habiendo una separación de 11 unidades entre ambos. En la figura 21 puede observarse la geometría descrita.

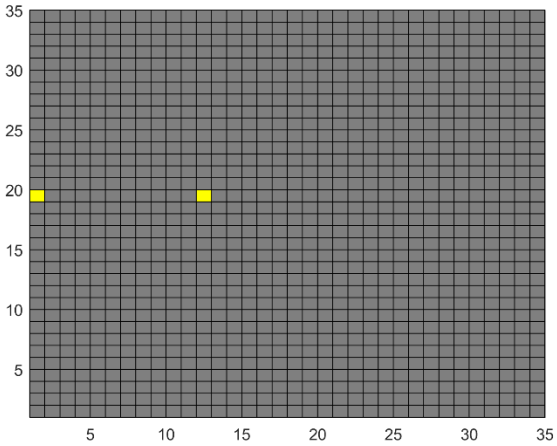


Figura 21: Malla para la segunda geometría.

En las figuras 22 y 23 se encuentran los resultados una vez aplicados los algoritmos Dijkstra y A*, respectivamente, al mallado anterior.

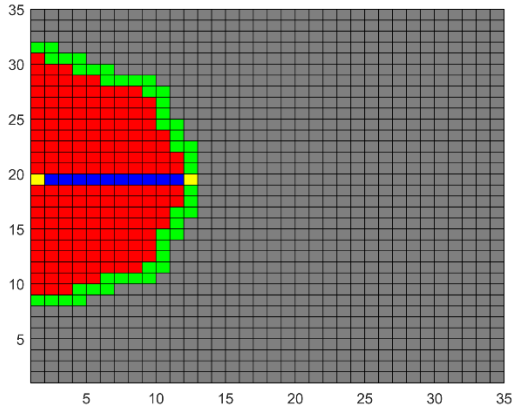


Figura 22: Camino óptimo encontrado aplicando Dijkstra a la segunda geometría.

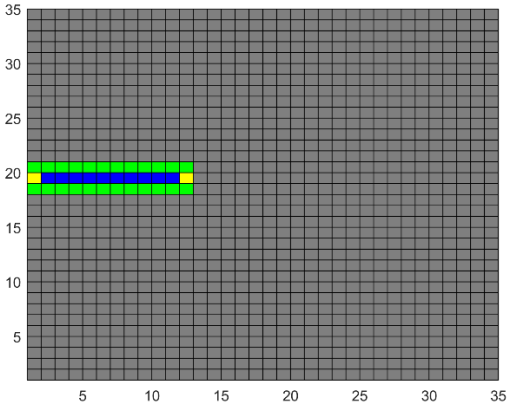


Figura 23: Camino óptimo encontrado aplicando A* a la segunda geometría.

3.1.7. Tercera geometría: Nodo origen en la esquina izquierda superior de la malla

El grafo utilizado será una malla rectangular sin obstáculos de 34x34 (nodos) donde la posición del nodo origen es [1,1] y la del nodo destino [1,12], habiendo una separación de 11 unidades entre ambos. En la figura 24 puede observarse la geometría descrita.

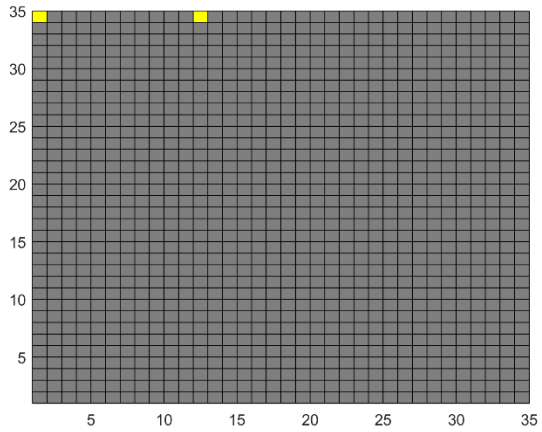


Figura 24: Malla para la tercera geometría.

En las figuras 25 y 26 se encuentran los resultados una vez aplicados los algoritmos Dijkstra y A*, respectivamente, al mallado anterior.

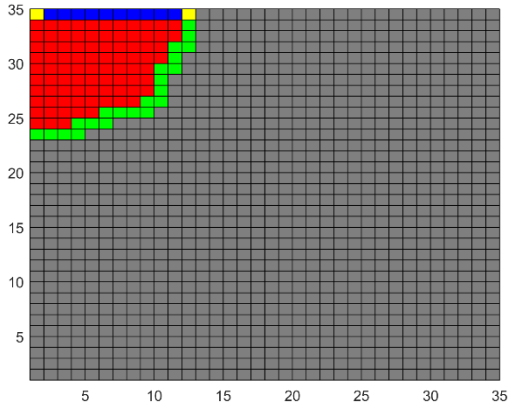


Figura 25: Camino óptimo encontrado aplicando Dijkstra a la tercera geometría.

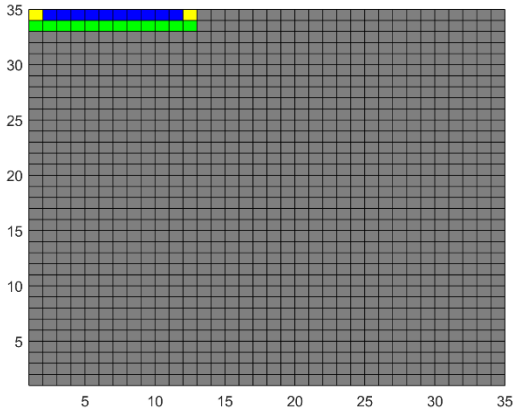


Figura 26: Camino óptimo encontrado aplicando A* a la tercera geometría.

3.1.8. Cuarta geometría: Obstáculo en L entre el nodo origen y el nodo destino

El grafo utilizado será una malla rectangular de 34x34 (nodos) donde la posición del nodo origen es [16,18] y la del nodo destino [16,29], habiendo una separación de 11 unidades entre ambos, y un obstáculo de 8 unidades de ancho y 8 de alto en forma de L, tal como se observa en la figura 27.

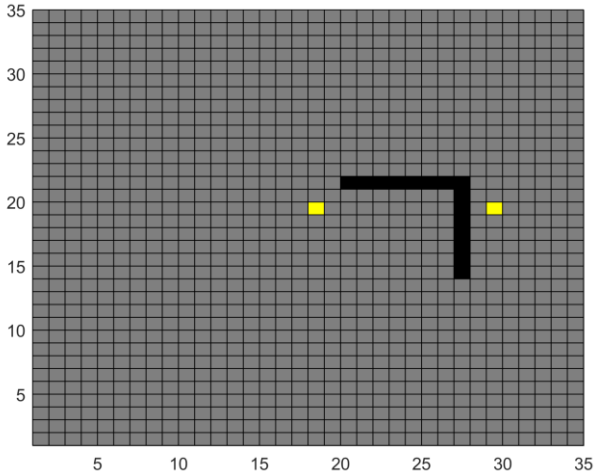


Figura 27: Malla para la cuarta geometría.

En las figuras 28 y 29 se encuentran los resultados una vez aplicados los algoritmos Dijkstra y A*, respectivamente, al mallado anterior.

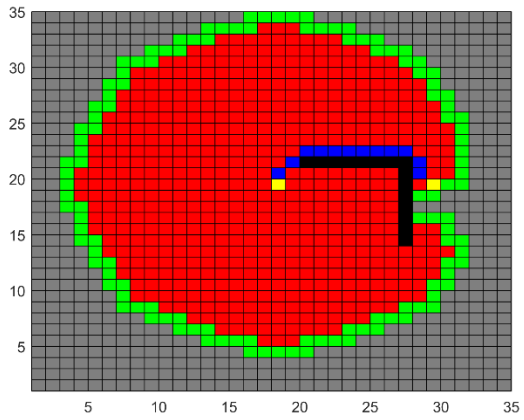


Figura 28: Camino óptimo encontrado aplicando Dijkstra a la cuarta geometría.

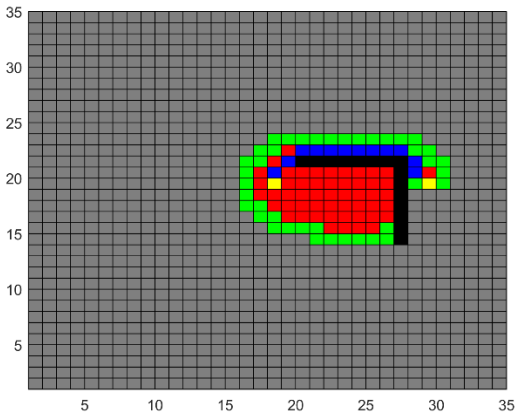


Figura 29: Camino óptimo encontrado aplicando A* a la cuarta geometría.

3.1.9. Quinta geometría: Obstáculo envolviendo al nodo destino

El grafo utilizado será una malla rectangular de 34x34 (nodos) donde la posición del nodo origen es [16,18] y la del nodo destino [16,29], habiendo una separación de 11 unidades entre ambos, y un obstáculo que envuelve al nodo destino permitiendo el paso por un único nodo, tal como se observa en la figura 30.

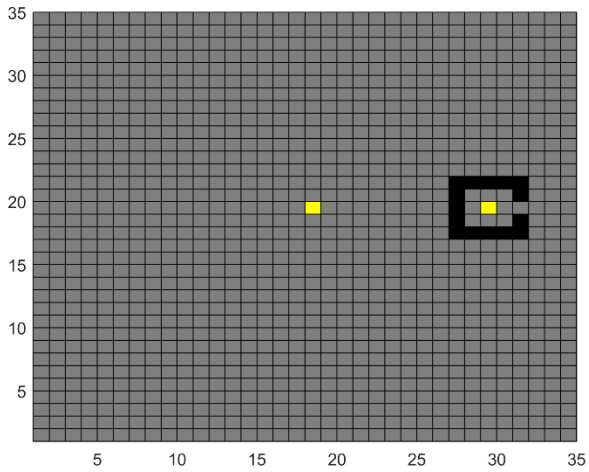


Figura 30 Malla para la quinta geometría.

En las figuras 31 y 32 se encuentran los resultados una vez aplicados los algoritmos Dijkstra y A*, respectivamente, al mallado anterior.

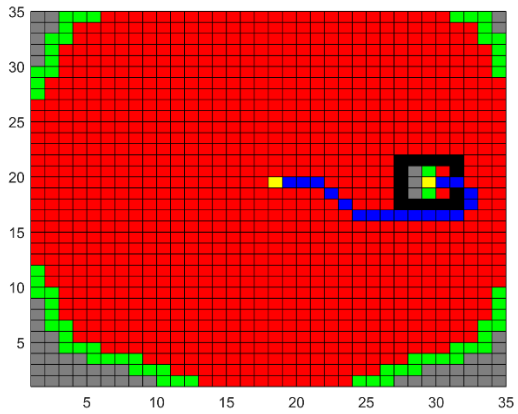


Figura 31: Camino óptimo encontrado aplicando Dijkstra a la quinta geometría.

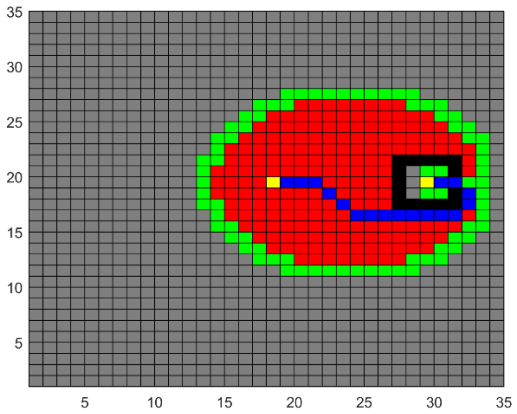


Figura 32: Camino óptimo encontrado aplicando A* a la quinta geometría.

3.1.10. Sexta geometría: Obstáculos envolviendo al nodo origen y al nodo destino

El grafo utilizado será una malla rectangular de 34x34 (nodos) donde la posición del nodo origen es [16,18] y la del nodo destino [16,29], habiendo una separación de 11 unidades entre ambos, y dos obstáculos que envuelven al nodo origen y al nodo destino permitiendo el paso por un único nodo, tal como se observa en la figura 33.

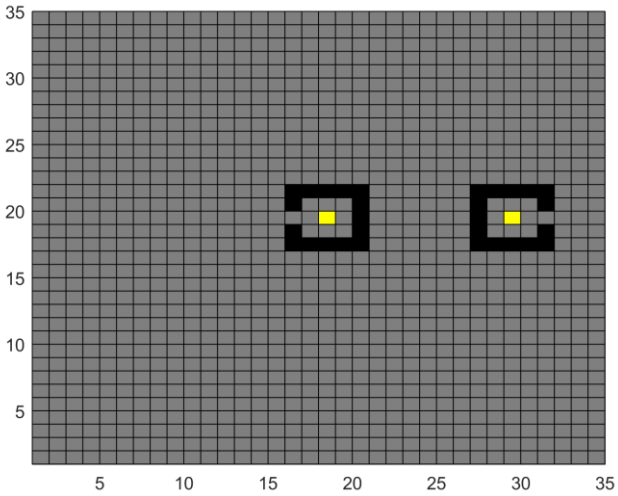


Figura 33: Malla para la sexta geometría.

En las figuras 34 y 35 se encuentran los resultados una vez aplicados los algoritmos Dijkstra y A*, respectivamente, al mallado anterior.

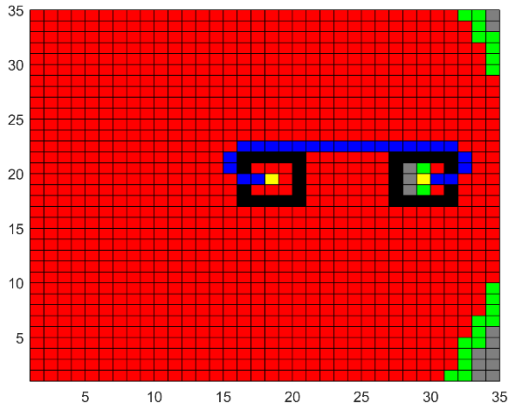


Figura 34: Camino óptimo encontrado aplicando Dijkstra a la sexta geometría.

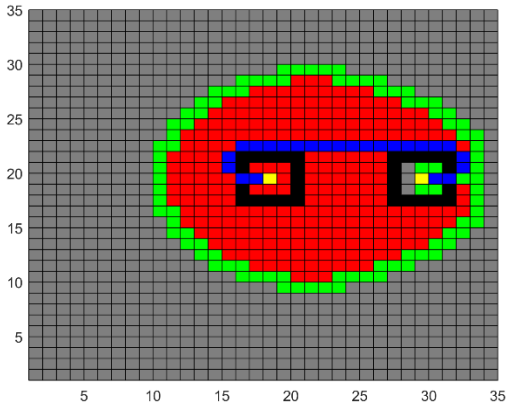


Figura 35: Camino óptimo encontrado aplicando A* a la sexta geometría.

3.2. Comparación algoritmos Dijkstra y A* en simulación de rutas marítimas

El programa SIMROUTE ha sido utilizado para comparar el rendimiento de los algoritmos Dijkstra y A* presentados en el apartado anterior cuando son utilizados en simulaciones de rutas marítimas con datos reales.

El algoritmo y metodología presentados en Métodos han sido aplicados a una ruta marítima de corta duración y a una ruta de larga duración. La resolución horizontal del mallado ha sido establecida en 16 direcciones por nodo y se ha considerado una velocidad del navío de 20 nudos. Han sido utilizados los datos del oleaje de los siguientes días 19/12/2016, 20/12/2016, 11/01/2017, 17/01/2017 y 06/02/2017 para la ruta Barcelona – Palma de Mallorca (ruta corta) y el oleaje de los días 06/02/2017, 07/02/2017, 08/02/2017 y 09/02/2017; 17/01/2017, 18/01/2017, 19/01/2017 y

20/01/2017; 19/01/2017, 20/01/2017, 21/01/2017 y 22/01/2017 para la ruta Peñíscola (España) – Chania (Grecia) (ruta larga).

En la tabla 3 se recogen los datos obtenidos una vez realizada las simulaciones para la ruta corta para ambos algoritmos. En la tabla 4 se recogen los valores para la simulación de la ruta larga para ambos algoritmos.

Tabla 3: Datos obtenidos de la comparación de los algoritmos Dijkstra y A* para una ruta de corta duración simulada mediante el SMROUTE.

Ruta corta				
Día	**Tiempo D (s)**	**Tiempo A (s)**	**ΔT (s)**	**Porcentaje de tiempo reducido**
19/12/2016	12,7	3,8	-8,9	70,4%
20/12/2016	13,3	4,5	-8,9	66,6%
11/01/2017	13,2	3,9	-9,4	70,7%
17/01/2017	12,4	4,0	-8,4	67,9%
06/02/2017	13,9	4,9	-9,0	64,7%

Tabla 4: Datos obtenidos de la comparación de los algoritmos Dijkstra y A* para una ruta de larga duración simulada mediante el SIMROUTE

Ruta larga				
Días	**Tiempo D (s)**	**Tiempo A (s)**	**ΔT (s)**	**Porcentaje de tiempo reducido**
06/02/2017 07/02/2017 08/02/2017 09/02/2017	1602,9	807,4	-795,4	49,6%
17/01/2017 18/01/2017 19/01/2017 20/01/2017	1641,3	726,5	-914,8	55,7%
19/01/2017 20/01/2017 21/01/2017 22/01/2017	1495,8	539,6	-956,2	63,9%

3.3. Aplicación del SIMROUTE

El programa SIMROUTE, descrito en el apartado anterior ha sido aplicado a 2 escenarios distintos: una ruta corta (duración menor a 1 día) y a una ruta larga (duración de 3 días o superior).

3.3.1. Ruta corta: Barcelona – Palma de Mallorca

El algoritmo y metodología presentados en Métodos han sido aplicados a la ruta marítima corta: Barcelona – Palma de Mallorca. La resolución horizontal del mallado ha sido establecida en 16 direcciones por nodo y se ha considerado una velocidad del navío de 20 nudos, velocidad próxima a la velocidad real de los barcos que realizan esta ruta diariamente. Han sido utilizados los datos del oleaje de los siguientes días: 19/12/2016, 20/12/2016, 21/12/2016, 09/01/2017, 11/01/2017, 13/01/2017, 17/01/2017, 18/01/2017, 19/01/2017, 20/01/2017, 21/01/2017, 22/01/2017, 05/02/2017, 06/02/2017, 04/03/2017. En ciertos días se han realizado más de una simulación con diferentes horas de salida.

En la tabla 5 se recogen los datos obtenidos una vez realizada las simulaciones en los días seleccionados. Los resultados para la ruta de mínima distancia sin considerar efectos del oleaje se encuentran en la tabla 6.

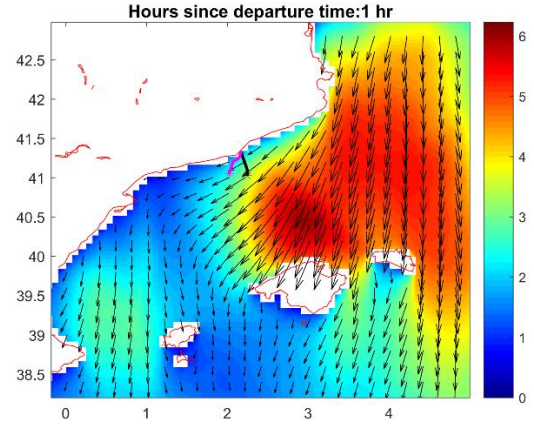
Tabla 5: Datos obtenidos de las simulaciones de la ruta corta para los casos de optimización y ruta fija WE.

RUTA CORTA BCN - PM					
Día	Hora salida	Hs_max (m)	T_opt (h)	T_fix_we (h)	lenght_opt (millas)
19/12/2016	0:00	3,2	7,00	7,00	133,292
19/12/2016	5:00	3,7	7,12	7,12	133,293
20/12/2016	6:00	5,0	8,28	8,79	138,859
20/12/2016	10:00	4,3	7,82	8,21	136,324
21/12/2016	0:00	3,4	7,26	7,34	134,159
21/12/2016	5:00	3,7	7,33	7,44	134,161
09/01/2017	0:00	3,0	6,99	7,07	134,163
09/01/2017	3:00	3,3	7,09	7,24	134,446
11/01/2017	0:00	4,0	7,23	7,41	134,729
11/01/2017	2:00	4,3	7,39	7,74	135,977
13/01/2017	0:00	1,9	6,79	6,80	133,311
17/01/2017	0:00	4,5	7,60	8,14	135,694
18/01/2017	0:00	3,9	7,26	7,40	133,594
19/01/2017	6:00	4,0	7,30	7,34	133,311
20/01/2017	0:00	3,8	7,28	7,29	133,293
21/01/2017	6:00	5,6	7,89	7,99	133,311
21/01/2017	11:00	6,8	8,94	9,33	133,596
22/01/2017	0:00	6,8	9,06	9,18	134,644
22/01/2017	3:00	6,0	8,43	8,53	133,878
22/01/2017	6:00	5,8	8,13	8,23	133,877
05/02/2017	0:00	3,1	6,97	7,01	133,304
06/02/2017	0:00	5,4	8,56	8,69	134,643
04/03/2017	0:00	6,2	8,12	8,31	136,861
04/03/2017	2:00	5,3	8,18	8,41	136,902
04/03/2017	4:00	4,9	7,97	8,26	136,624

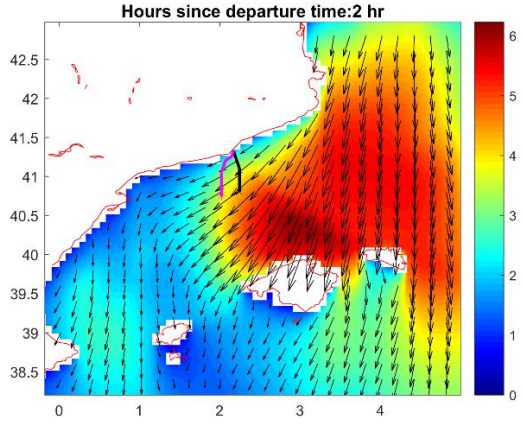
Tabla 6: Datos obtenidos en la ruta fija a una velocidad de crucero de 20 nudos.

RUTA CORTA FIJA BCN - PM	
T_fix (h)	lenght_fix (millas)
6,665	133,292

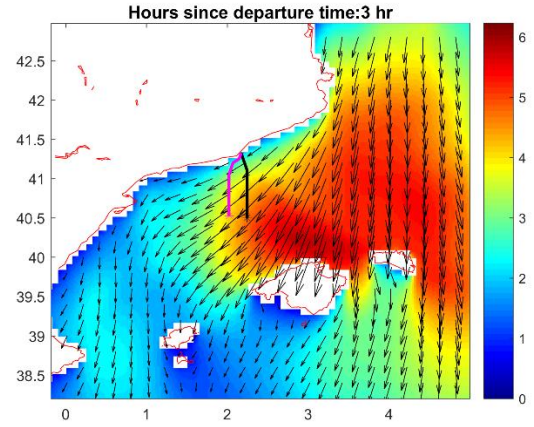
En la figura 36 se puede observar la evolución temporal, para el día 17/01/2017 con hora de salida las 00:00h, de la ruta optimizada, de color rosa, y de la ruta de distancia mínima considerando los efectos del oleaje, de color negro, desde su salida del puerto de Barcelona hasta su llegada al puerto de Palma de Mallorca.



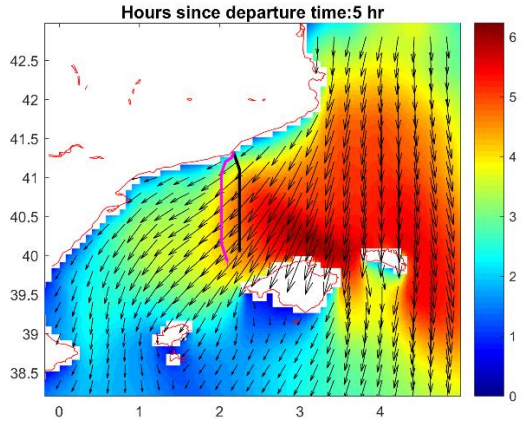
a) Avance de la ruta fija, la negra, y la ruta optimizada, la rosa, 1 h después de la salida.



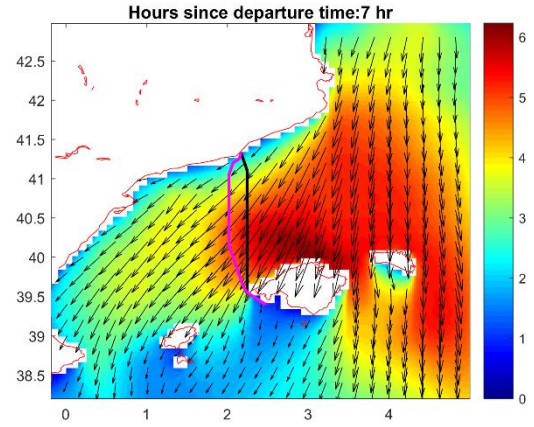
b) Avance de la ruta fija, la negra, y la ruta optimizada, la rosa, 2 h después de la salida.



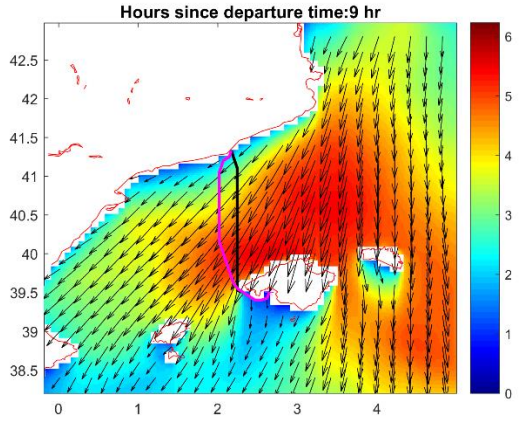
c) Avance de la ruta fija, la negra, y la ruta optimizada, la rosa, 3 h después de la salida.



d) Avance de la ruta fija, la negra, y la ruta optimizada, la rosa, 5 h después de la salida.



e) Avance de la ruta fija, la negra, y la ruta optimizada, la rosa, 7 h después de la salida.



f) Llegada al puerto de destino de la ruta fija, la negra y la ruta optimizada, la rosa.

Figura 36: Evolución temporal de las rutas fija y optimizada el día 17/01/2017 iniciando el viaje a las 00:00h.

3.3.2. Ruta larga: España - Grecia

El algoritmo y metodología presentados en Métodos han sido aplicados a la ruta marítima larga: Peñíscola (España) – Chania (Grecia). La resolución horizontal del mallado ha sido establecida en

16 direcciones por nodo y se ha considerado una velocidad del navío de 20 nudos. Han sido utilizados los datos del oleaje de los siguientes días: 06/02/2017, 07/02/2017, 08/02/2017 y 09/02/2017.

En la tabla 7 se recogen los datos obtenidos al realizar la simulación durante los días seleccionados, contiene los resultados para la ruta optimizada, la ruta de distancia mínima considerando los efectos del oleaje y sin considerar los efectos del oleaje.

Tabla 7: Datos obtenidos de la simulación de la ruta larga para los casos de optimización, ruta fija con efectos del oleaje y ruta fija sin efectos del oleaje.

RUTA LARGA ES - GR			
Días	**T_opt (h)**	**T_fix_we (h)**	**lenght_opt (millas)**
06/02/2017 07/02/2017 08/02/2017 09/02/2017	69,44	72,82	1.195,067
T_fix (h)		**lenght_fix (millas)**	
58,617		1.172,331	

En la figura 37 se puede observar las trayectorias descritas por la ruta de distancia mínima, en verde, y la ruta optimizada, en morado.

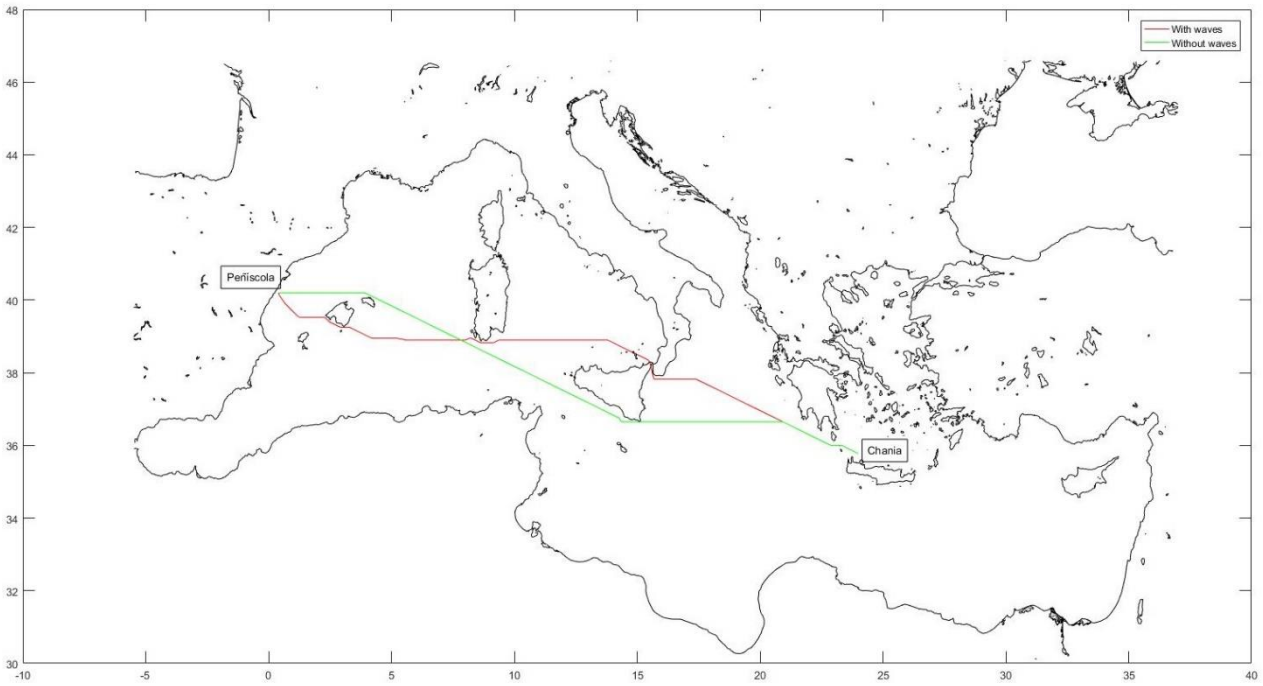


Figura 37: Trayecto de la ruta de distancia mínima, en verde, y la ruta una vez optimizada, en morado, con salida del puerto de Peñíscola a las 00:00h del día 06/02/2017.

3.4. Impactos anuales

Acorde a la metodología explicada en Métodos se ha considerado dos cotas de altura de ola significante a tener en cuenta, la de 3 m y la de 4 m. En las tablas 8, 9, 10, 11 y 12 se recogen los

días en que se superó la cota de 3 m para los años 2013, 2014, 2015, 2016 y 2017*. Dentro de las mismas tablas está también separados los días que se superó la cota de 4 m de altura de ola significativa, siendo estos los primeros en aparecer y a continuación todos los comprendidos entre 3 m y 4 m de altura de ola significativa.

(*) Del año 2017 solo se dispone de los datos de medio año, por lo que solo está recogido en las tablas los datos del 1 de enero de 2017 al 30 de junio de 2017. Al ser la época de tormenta a principios de año y a finales, siendo todo el verano por lo general muy calmado y sin tormentas considerables, se podrá comparar también los resultados del año 2017 con los de los demás años.

Tabla 8: Días del año 2013 en que se superó los 3m de altura significativa de ola en el recorrido de la ruta.

AÑO 2013				
Día	SIMAR (2110118) Hs máxima (m)	SIMAR (2110126) Hs máxima (m)	SIMAR (2110132) Hs máxima (m)	Hs máxima en la ruta
03-ene	4,0	3,7	-	4,0
19-ene.	4,4	4,5	4,1	4,5
20-ene.	4,4	3,8	-	4,4
23-ene.	4,3	4,0	-	4,3
2-feb.	5,0	4,3	-	5,0
3-feb.	4,8	4,5	3,4	4,8
7-feb.	4,4	3,4	-	4,4
1-mar.	4,5	5,3	4,7	5,3
13-mar.	4,5	3,8	-	4,5
14-mar	4,9	4,2	3,1	4,9
28-abr	3,1	4,2	4,4	4,4
29-abr	-	4,0	4,1	4,1
11-nov	4,0	3,6	-	4,0
12-nov	4,4	3,5	-	4,4
16-nov	3,2	3,2	5,3	5,3
19-nov	5,3	5,0	4,1	5,3
26-nov	4,2	3,8	-	4,2
27-nov	4,3	4,5	3,8	4,5
01-dic	5,8	5,6	4,7	5,8
20-dic	5,5	4,5	-	5,5
25-dic	4,9	4,4	3,8	4,9
26-dic	4,3	4,5	3,9	4,5
15-ene	3,1	-	-	3,1
17-ene	3,1	-	-	3,1
24-ene	3,7	-	-	3,7
25-ene	3,2	-	-	3,2
28-ene	3,7	3,2	-	3,7
08-feb	3,0	-	-	3,0
10-feb	3,5	-	-	3,5
11-feb	-	3,6	3,3	3,6
12-feb	3,2	3,2	-	3,2
23-feb	3,8	3,3	-	3,8
24-feb	3,8	3,1	-	3,8
28-feb	-	3,5	3,9	3,9
02-mar	-	-	3,2	3,2
06-mar	3,2	3,4	3,2	3,4
13-mar	-	3,8	-	3,8
15-mar	3,5	-	-	3,5
26-mar	3,1	3,3	-	3,3
30-mar	3,1	3,2	-	3,2
20-abr	3,2	-	-	3,2
25-abr	3,3	3,6	3,4	3,6
29-oct	3,3	-	-	3,3
30-oct	3,8	3,2	-	3,8
06-nov	3,2	-	-	3,2
10-nov	-	3,3	-	3,3
14-nov	3,2	3,1	-	3,2
15-nov	3,6	-	-	3,6
17-nov	-	-	3,1	3,1
18-nov	-	-	3,3	3,3
20-nov	3,7	3,2	-	3,7
22-nov	3,6	3,3	-	3,6
25-nov	3,1	-	-	3,1
28-nov	3,1	-	-	3,1
02-dic	3,3	-	-	3,3
03-dic	3,2	3,3	3,1	3,3
20-dic	-	-	3,2	3,2
24-dic	3,2	3,5	3,2	3,5

Tabla 9: Días del año 2014 en que se superó los 3m de altura significativa de ola en el recorrido de la ruta.

AÑO 2014				
Día	SIMAR (2110118) Hs máxima (m)	SIMAR (2110126) Hs máxima (m)	SIMAR (2110132) Hs máxima (m)	Hs máxima en la ruta
5-ene.	4,7	5,0	3,4	5,0
10-feb.	5,5	4,5	3,0	5,5
4-mar.	5,3	4,7	3,4	5,3
4-abr.	4,2	3,1	-	4,2
4-nov.	4,5	3,6	-	4,5
9-dic.	6,0	5,5	4,3	6,0
10-dic.	4,7	-	-	4,7
29-dic.	4,8	4,8	3,5	4,8
30-dic.	4,0	3,8	-	4,0
4-ene.	3,5	-	-	3,5
20-ene.	3,5	-	-	3,5
21-ene.	3,2	-	-	3,2
28-ene.	3,2	3,3	-	3,3
29-ene.	3,6	-	-	3,6
2-feb.	3,3	-	-	3,3
5-feb.	3,3	3,4	-	3,4
8-feb.	3,9	3,7	3,4	3,9
11-feb.	-	3,9	-	3,9
18-feb.	3,4	3,5	-	3,5
28-feb.	3,9	-	-	3,9
1-mar.	3,9	3,3	-	3,9
3-mar.	3,9	3,7	-	3,9
25-mar.	3,4	-	-	3,4
26-mar.	3,6	-	-	3,6
22-oct.	3,6	3,5	-	3,6
30-nov.	-	-	3,4	3,4
1-dic.	-	-	3,1	3,1
2-dic.	3,4	3,6	-	3,6
28-dic.	3,6	3,9	-	3,9
31-dic.	3,2	-	-	3,2

Tabla 10: Días del año 2015 en que se superó los 3m de altura significativa de ola en el recorrido de la ruta.

AÑO 2015				
Día	SIMAR (2110118) Hs máxima (m)	SIMAR (2110126) Hs máxima (m)	SIMAR (2110132) Hs máxima (m)	Hs máxima en la ruta
25-ene.	6,3	5,9	4,5	6,3
26-ene.	4,2	3,7	-	4,2
31-ene.	4,8	4,6	-	4,8
1-feb.	4,7	4,0	3,4	4,7
4-feb.	4,9	4,2	3,3	4,9
5-feb.	4,8	4,5	3,1	4,8
6-feb.	4,1	3,8	-	4,1
9-feb.	4,1	3,8	3,2	4,1
17-feb.	4,9	4,5	3,6	4,9
18-feb.	4,7	4,5	3,5	4,7
24-feb.	4,8	4,0	-	4,8
5-mar.	5,4	4,9	3,9	5,4
6-mar.	4,8	4,3	-	4,8
25-mar.	4,4	4,2	3,0	4,4
30-sep.	4,9	5,2	4,9	5,2
21-nov.	5,2	4,2	-	5,2
27-nov.	4,7	4,8	3,7	4,8
1-ene.	-	3,0	-	3,0
24-ene.	3,3	3,0	-	3,3
28-ene.	3,1	-	-	3,1
30-ene.	3,7	3,7	3,7	3,7
3-feb.	3,9	3,3	-	3,9
22-feb.	3,2	-	-	3,2
25-feb.	3,8	3,1	-	3,8
27-feb.	3,7	3,1	-	3,7
28-feb.	3,5	3,1	-	3,5
20-mar.	3,2	3,3	3,7	3,7
21-mar.	-	-	3,6	3,6
26-mar.	3,6	3,2	-	3,6
15-may.	3,5	3,2	-	3,5
16-may.	3,3	-	-	3,3
1-oct.	-	-	3,4	3,4
2-nov.	3,2	3,8	3,9	3,9
22-nov.	3,8	-	-	3,8
26-nov.	3,9	-	-	3,9

Tabla 11: Días del año 2016 en que se superó los 3m de altura significativa de ola en el recorrido de la ruta.

AÑO 2016				
Día	SIMAR (2110118) Hs máxima (m)	SIMAR (2110126) Hs máxima (m)	SIMAR (2110132) Hs máxima (m)	Hs máxima en la ruta
11-ene	3,3	4,2	4,2	4,2
09-feb	4,1	3,6	-	4,1
14-feb	3,5	4,0	3,2	4,0
16-feb	5,1	-	4,1	5,1
29-feb	4,3	3,4	-	4,3
01-mar	4,1	3,1	-	4,1
20-dic	5,1	4,5	5,1	5,1
21-dic	4,0	3,7	3,1	4,0
04-ene	3,3	-	-	3,3
07-ene	3,6	-	-	3,6
10-ene	3,2	3,6	3,7	3,7
16-ene	3,5	3,1	-	3,5
04-feb	3,8	3,5	-	3,8
07-feb	3,5	3,4	-	3,5
10-feb	3,3	3,1	-	3,3
12-feb	3,2	-	-	3,2
13-feb	3,2	-	-	3,2
27-feb	3,4	3,2	-	3,4
03-mar	3,7	-	-	3,7
06-abr	3,3	-	-	3,3
14-nov	3,8	3,5	-	3,8
15-nov	3,9	3,6	-	3,9
21-nov	-	3,1	3,0	3,1
23-nov	3,1	-	-	3,1
19-dic	3,5	3,9	3,7	3,9
22-dic	3,4	3,7	-	3,7

Tabla 12: Días de la primera mitad del año 2017 en que se superó los 3m de altura significativa de ola en el recorrido de la ruta.

AÑO 2017*				
Día	SIMAR (2110118) Hs máxima (m)	SIMAR (2110126) Hs máxima (m)	SIMAR (2110132) Hs máxima (m)	Hs máxima en la ruta
11-ene	4,3	-	-	4,3
17-ene	5,2	5,0	-	5,2
18-ene	4,1	-	-	4,1
19-ene	-	4,2	4,1	4,2
20-ene	3,1	3,9	4,0	4,0
21-ene	6,7	7,1	6,0	7,1
22-ene	5,6	7,2	6,7	7,2
23-ene	-	4,2	4,0	4,2
05-feb	4,5	5,7	5,0	5,7
06-feb	6,2	6,3	4,5	6,3
04-mar	6,2	4,8	5,2	6,2
19-abr	-	4,2	4,0	4,2
05-ene	3,5	-	-	3,5
06-ene	3,1	-	-	3,1
09-ene	3,3	3,1	-	3,3
13-ene	3,7	3,3	-	3,7
15-ene	3,1	-	-	3,1
16-ene	3,0	-	-	3,0
04-feb	-	3,2	3,6	3,6
08-feb	3,4	-	-	3,4
09-feb	3,4	-	-	3,4
13-mar	3,4	3,7	3,1	3,7
25-mar	-	-	3,8	3,8
27-abr	3,1	3,1	-	3,1

En la tabla 13 se encuentra el porcentaje de viajes durante el año que se verían beneficiados por la optimalización de la ruta suponiendo las consideraciones expuestas en el apartado de Métodos sobre la estimación del número de viajes por año.

Tabla 13: Nº de viajes beneficiados por la aplicación de la optimalización de la ruta en función de la cota definida para el estudio. (* para el año 2017 solo se dispone de datos de la primera mitad año)

Impacto anual						
Año	Nº de días (4m)	Nº viajes (4m)	Porcentaje de viajes	Nº de días (3m)	Nº viajes (3m)	Porcentaje de viajes
2013	20	78	6,23%	58	202	16,13%
2014	9	34	2,72%	30	110	8,79%
2015	17	58	4,63%	35	114	9,11%
2016	8	32	2,55%	26	96	7,66%
2017*	12	42	6,75%	24	82	13,18%

4. Discusión y conclusiones

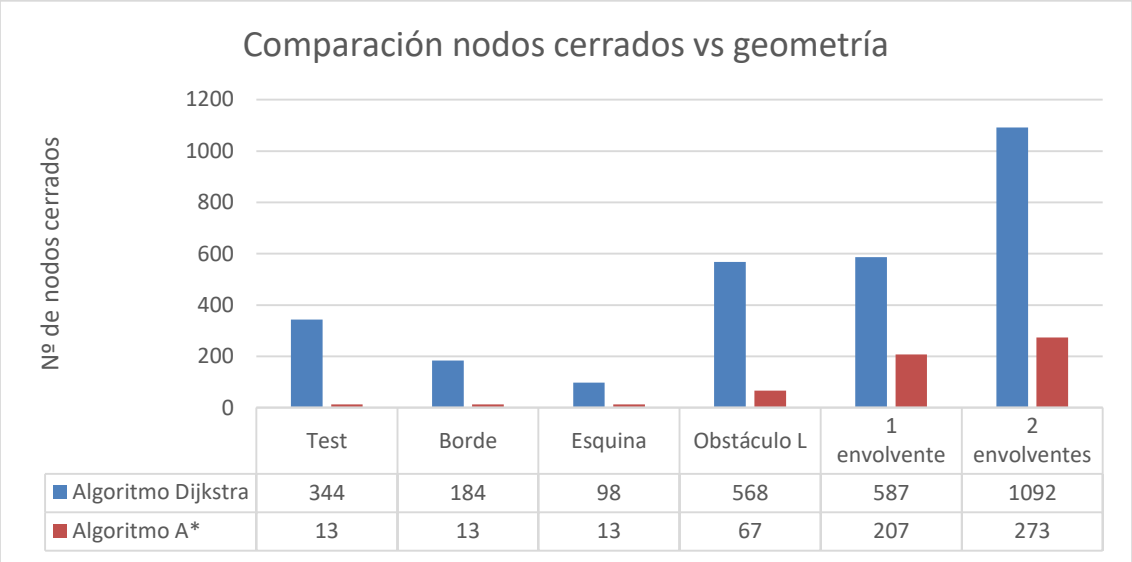
Este trabajo centrado en los algoritmos Dijkstra y A* ha llevado a cabo una comparación académica de ambos ante las mismas geometrías definidas y otra más práctica al comparar los algoritmos integrados en el programa para el cálculo de la optimización de rutas marítimas. Además, se ha realizado una serie de simulaciones en rutas marítimas de las que se va a buscar una relación entre los parámetros del oleaje y sus efectos en los cambios en la ruta de distancia mínima.

4.1. Dijkstra y A*

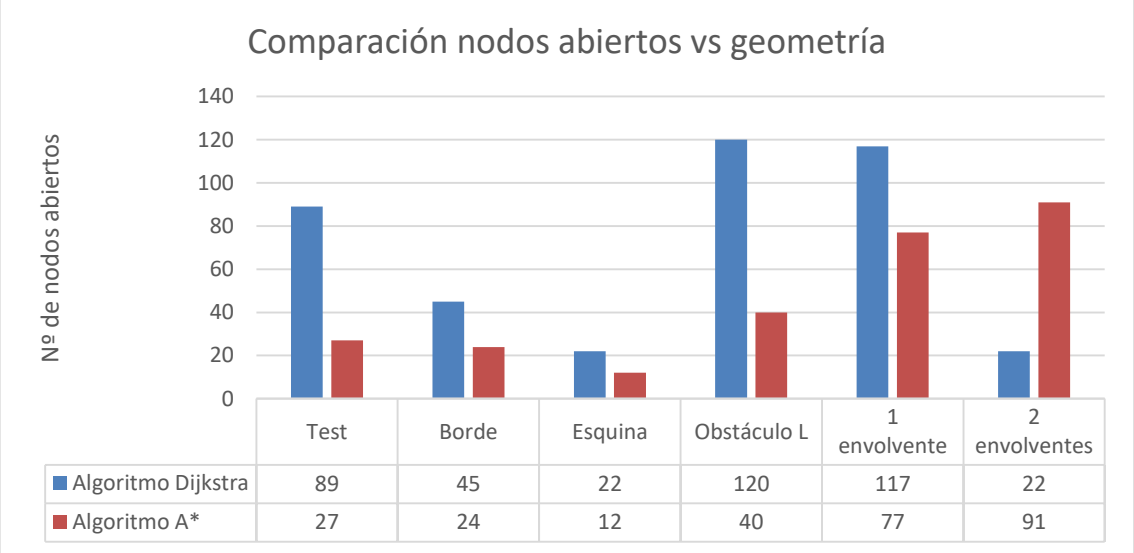
Como primer paso a la comparación de ambos algoritmos se ha simulado el caso más sencillo posible y se han ido introduciendo progresivamente obstáculos para observar el comportamiento de ambos algoritmos ante estos cambios. Además de las geometrías estáticas, se ha intentado buscar respuesta a las preguntas: ¿A qué ratio crecen los nodos visitados y abiertos al recorrer distancias mucho mayores?, ¿Cómo evoluciona el tiempo de computación para grandes distancias?, ¿Cómo afecta el tamaño del obstáculo a los algoritmos? Por lo que se han realizado los dos casos de aumento lineal, en el primero de la distancia de separación entre nodos origen y destino y en el segundo caso en el tamaño de un obstáculo.

Habiendo realizado estos cálculos se ha podido observar que el algoritmo Dijkstra recorre muchos más nodos que el A*, al utilizar el primero una técnica voraz acaba haciendo un barrido de muchos nodos antes de llegar a encontrar el camino óptimo. A* gracias a la función heurística es capaz de encontrar el camino óptimo con menos iteraciones y visitando menos nodos durante el proceso respecto al algoritmo Dijkstra. Esto queda reflejado sobretodo en sistemas con grandes cantidades de nodos, como en el apartado 3.1.3 de aumento lineal de la distancia de separación entre origen y destino, donde el tiempo de computación del algoritmo Dijkstra se incrementaba exponencialmente a medida que aumentábamos la distancia entre los nodos mientras que para A*, al contar con la función heurística, en este caso no se inmutaba al no tener obstaculizado la recta que unía ambos puntos. En el caso del apartado 3.1.4 de aumento lineal del tamaño del obstáculo vertical el algoritmo A* sí que se veía afectado y sufría un aumento del tiempo de computación, pero siendo este siempre menor al tiempo de computación del Dijkstra.

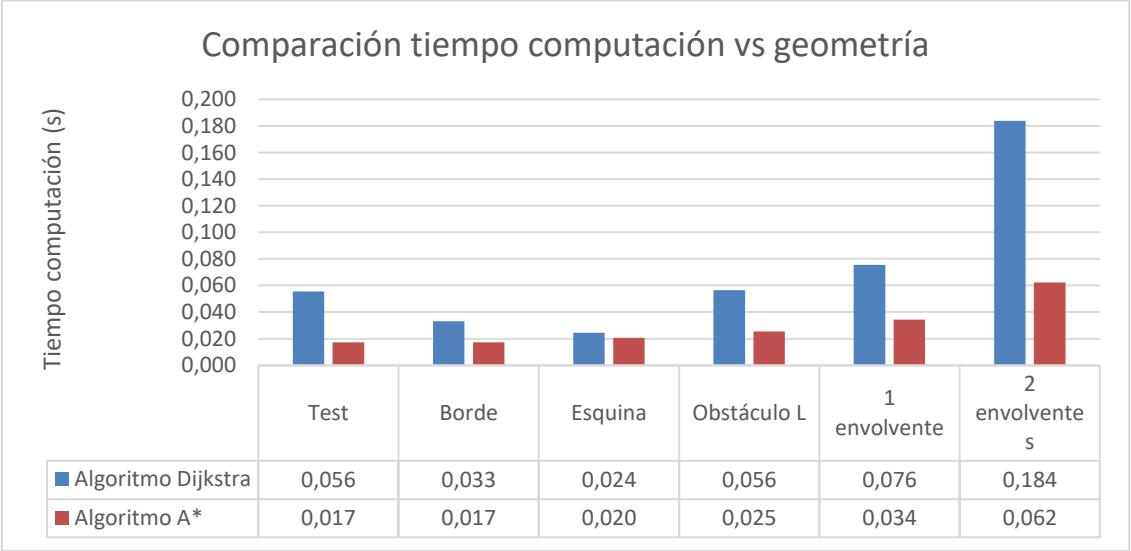
En la figura 38 se puede observar la comparación de los resultados obtenidos en las 6 geometrías calculadas. Se comprueba que el algoritmo Dijkstra aumenta su tiempo de computación a mayor complejidad, al visitar más nodos, siendo los casos de tener el nodo origen al borde de la malla o en una de las esquinas beneficioso para el algoritmo Dijkstra al tener restringidas una o dos direcciones por las que expandirse, visitando así menos nodos al encontrar el camino óptimo



a) Comparación del nº de nodos cerrados en función de la geometría estudiada



b) Comparación del nº de nodos abiertos en función de la geometría estudiada



c) Comparación del tiempo de computación en función de la geometría estudiada

Figura 38: Comparación de los resultados calculados en las 6 geometrías definidas en el capitulo3.

El algoritmo A* fue creado con la finalidad de mejorar el algoritmo Dijkstra y como se ha podido comprobar, en cuanto a número de nodos visitados y tiempo de computación se trata de un algoritmo más eficiente, por lo que esa mejora buscada se consiguió.

Estas diferencias mencionadas se ven reflejadas cuando los algoritmos son implementados como parte de un sistema mayor, como en el caso del programa SIMROUTE para la simulación de optimización de rutas marítimas. Tal como se ha mostrado en las tablas 3 y 4 el hecho de utilizar el algoritmo A* frente al Dijkstra se traduce en una reducción generalmente superior al 50% de los tiempos de computación que tarda el programa en ejecutarse, siendo esto especialmente útil en el caso de las rutas de larga duración que al manejar muchos más datos y trabajando en una malla de mayor cantidad la reducción es superior a 10 minutos. En caso de que el programa trabajara en rutas mucho mayores, por ejemplo, por el océano Atlántico o el océano Pacífico donde las distancias entre dos puertos distantes pueden llegar a ser 3 y 4 veces mayores que dentro del mar Mediterráneo la reducción podría alcanzar a ser mayor a 1h.

Como se ha evidenciado en este trabajo, el Dijkstra es un algoritmo útil por su simpleza y su fácil implementación, pero para implementaciones prácticas donde se maneje una gran cantidad de datos conlleva grandes tiempos de computación. Debido a eso y al hecho que el Dijkstra solo funciona con una función objetivo a optimizar, en la actualidad y como se ha mencionado en la introducción hay otra serie de algoritmos que son utilizados en herramientas comerciales de modificación de rutas. Además, con el afán de mejorar el rendimiento del Dijkstra se han creado mejoras del algoritmo que son mucho más veloces que el original y encuentran el camino óptimo más velozmente, como puede ser la variación recurrente del algoritmo Dijkstra, el cual evalúa todos los nodos de la frontera a la vez, necesitando muchas menos iteraciones para llegar al nodo destino. A*, como se ha visto en este trabajo es una mejora interesante para esos sistemas que estén basados en el algoritmo Dijkstra al permitir su función heurística reducir los costes de computación.

4.2. Optimización de rutas marítimas

Después de haber realizado la comparación entre los algoritmos Dijkstra y A* implementados en el SIMROUTE y al comprobar que el A* tiene unos tiempos de computación mucho menor se han realizado las simulaciones venideras usando solo el algoritmo A*. De estas simulaciones se pretende estudiar el efecto del oleaje en la duración del viaje y en el trazado de la ruta optimizada. También se han realizado los cálculos referentes al estudio estadístico del efecto que tendría aplicar el SIMROUTE en los viajes realizados en el trayecto Barcelona – Palma de Mallorca en años pasados.

4.2.1. Efecto del oleaje en la duración del viaje

En total se han realizado 25 simulaciones distintas, algunas pertenecientes al mismo día, pero en horas de salida diferentes. La primera comparación realizada es entre la ruta de distancia mínima sin considerar efectos del oleaje y la ruta de distancia mínima considerando los efectos del oleaje. En la figura 39 se puede observar la relación entre la altura de ola significante y el incremento de tiempo que comporta considerar o no el efecto del oleaje.

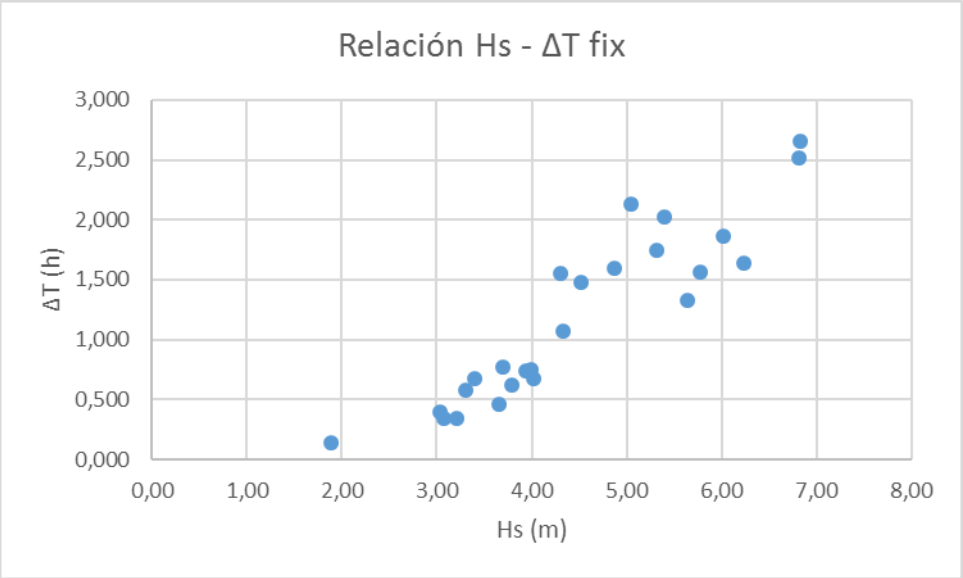


Figura 39: Comparación entre la ruta de distancia mínima para las 25 simulaciones relacionando la altura significativa de ola con el tiempo de más necesario al considerar el efecto del oleaje.

Se puede observar una relación lineal entre la altura de ola significante y el incremento de tiempo en la ruta de distancia mínima si tenemos en cuenta o no el efecto del oleaje. Para los datos obtenidos, para oleajes entre 3 y 4 m de altura de ola significante el tiempo real respecto al teórico si no considerásemos el oleaje del mar aumenta aproximadamente entre media y una hora. En cambio, para alturas de ola significante mayores a 4m este incremento ya es mayor a 1h, llegando a tener algunos incrementos superiores a las dos horas y media, incremento que supone más de un 35% de la duración si no consideramos el oleaje.

Una vez vista la comparación entre considerar o no el efecto del oleaje, ahora se estudiará el efecto de este entre la ruta optimizada y la ruta de distancia mínima considerando el efecto del oleaje. En la figura 40 se puede observar cómo afecta la altura de ola significante al incremento de tiempo de la ruta optimizada respecto a la ruta de distancia mínima considerando efectos del oleaje.

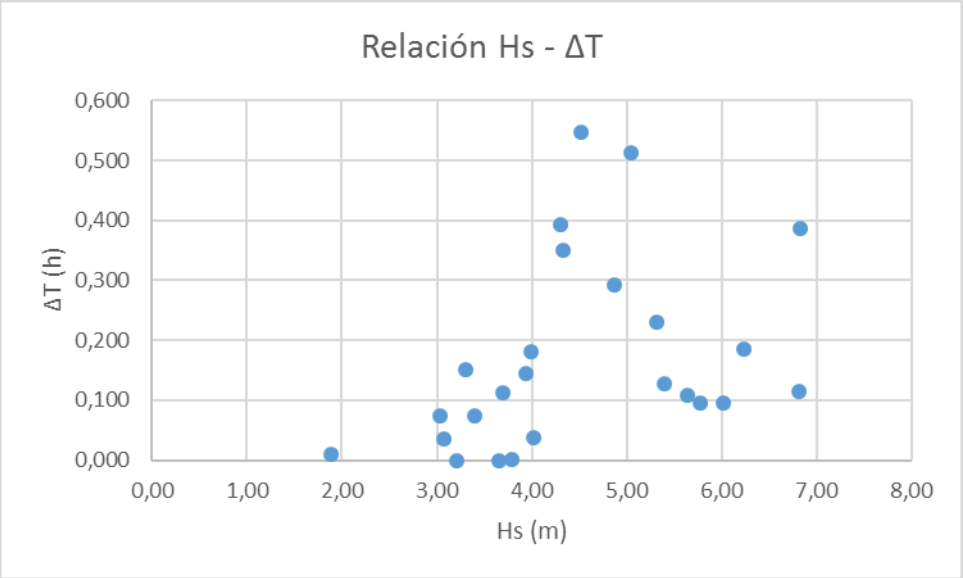
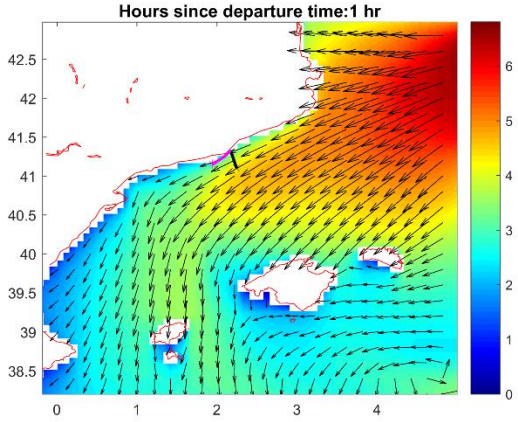


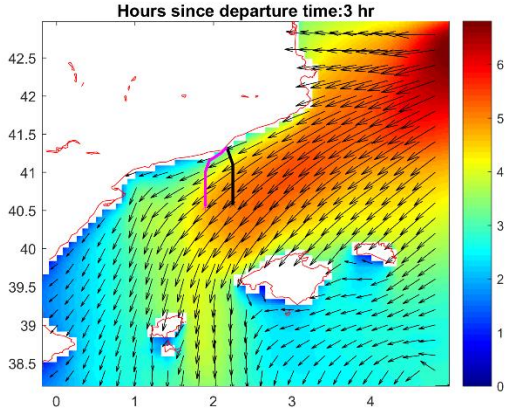
Figura 40: Comparación entre la ruta de distancia mínima considerando los efectos del oleaje y la ruta optimizada para las 25 simulaciones relacionando la altura significativa de ola con la reducción de tiempo conseguida.

Este caso ya no es tan claro como el anterior, las alturas de ola significante que producen una mayor reducción del tiempo de viaje no son las mayores sino, en este caso, las comprendidas entre 4 y 5m. Esto puede indicar que, en este caso, es necesario tener en cuenta más parámetros para entender la relación entre estos dos.

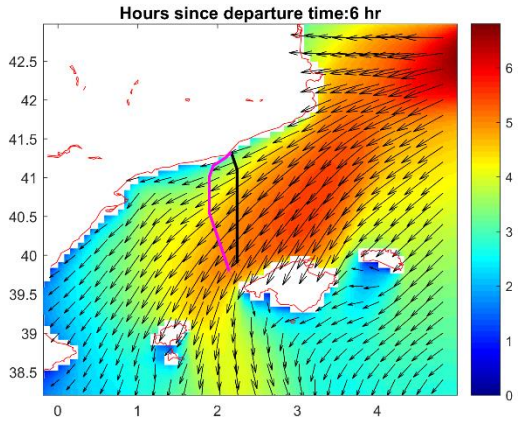
Con tal de profundizar más en este fenómeno en que las alturas mayores no tienen por qué implicar una mayor reducción del tiempo de viaje nos concentraremos en dos casos particulares, uno donde la altura de ola significativa está comprendida entre los 4m y 5m y otro donde es mayor de 6m. En la figura 41 puede observarse la evolución del viaje del día 20/12/2016, iniciado a las 6:00 h, en el que la altura de ola significante máxima fue de 5,05 m y la ruta tuvo una reducción del tiempo de 0,51 h y un cambio del trazado de 5,57 millas. Del mismo modo en la figura 42 puede observarse la evolución del viaje esta vez del día 22/01/2017, iniciado a las 3:00h, en el que la altura significante máxima del oleaje fue de 6,01m y la ruta tuvo una reducción del tiempo de 0,1h y un cambio del trazado de 0,59 millas, bastante menor que el primer suceso.



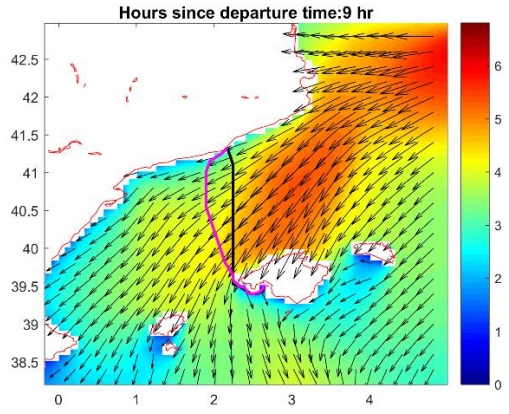
a) Avance de la ruta fija, la negra, y la ruta optimizada, la rosa, 1 h después de la salida.



b Avance de la ruta fija, la negra, y la ruta optimizada, la rosa, 3 h después de la salida.

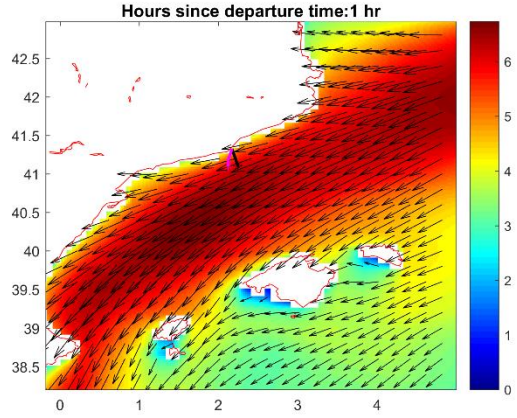


c) Avance de la ruta fija, la negra, y la ruta optimizada, la rosa, 6 h después de la salida..

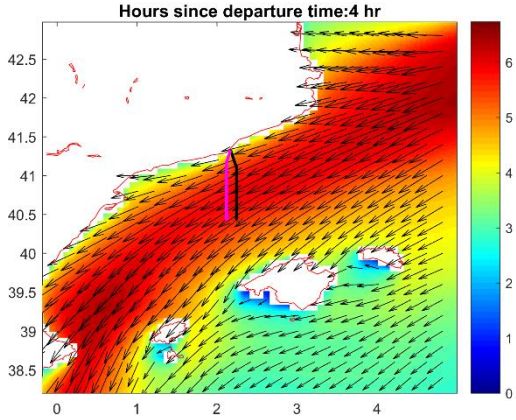


d) Llegada al puerto de destino de la ruta fija, la negra y la ruta optimizada, la rosa.

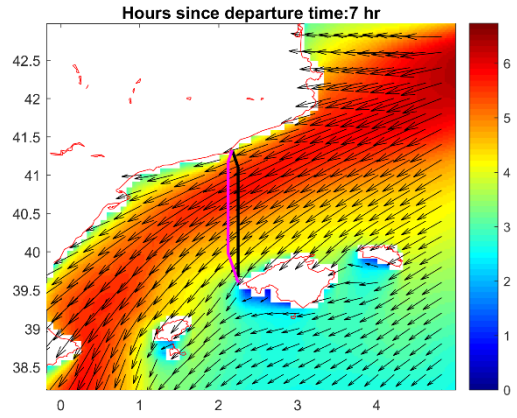
Figura 41: Evolución de las rutas fija y optimizada el día 20/12/2016 iniciando el viaje a las 6:00h.



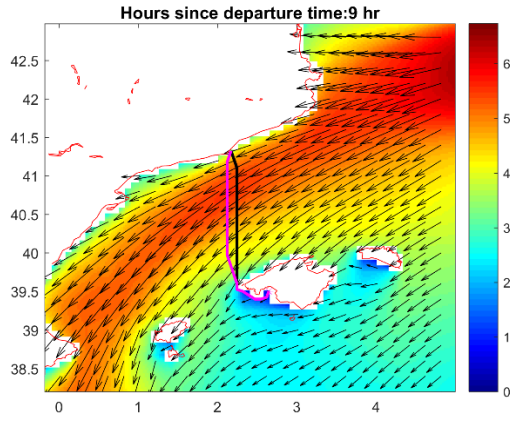
a) Avance de la ruta fija, la negra, y la ruta optimizada, la rosa, 1 h después de la salida.



b) Avance de la ruta fija, la negra, y la ruta optimizada, la rosa, 4 h después de la salida.



c) Avance de la ruta fija, la negra, y la ruta optimizada, la rosa, 7 h después de la salida.



d) Llegada al puerto de destino de la ruta fija, la negra y la ruta optimizada, la rosa.

Figura 42: Evolución de las rutas fija y optimizada el día 22/01/2017 iniciando el viaje a las 3:00h.

La diferencia principal que puede observarse entre las figuras 41 y 42, es que el día 22/01/2017 la tormenta es bastante uniforme, presentando grandes alturas de ola, pero abarcando un gran ancho de millas en el mar y manteniendo la misma dirección relativa del oleaje, por lo que a pesar de realizar grandes cambios en el trazado de la ruta el navío enfrentaría la misma altura significativa de ola, por lo que no habría beneficio en hacerlo. En cambio, el día 20/12/2016, la tormenta está presente mayoritariamente solo por uno de los lados de la ruta y la evolución espacial de la tormenta durante los instantes en que se realiza el viaje permite que al hacer un cambio de trazado el navío ha de enfrentarse a alturas de ola menores que las presentes en la ruta de distancia mínima.

Por tanto, deducimos con este ejemplo que no solo es importante la dirección y altura del oleaje sino también el patrón espacial de la evolución de la tormenta.

4.2.2. Efecto del oleaje en el trazado de la ruta óptima

Una vez analizado los efectos en la reducción del tiempo de viaje, ahora se enfocará el análisis en el cambio del trazado de la ruta que realiza la optimización respeto a la ruta de mínima distancia. A mayor incremento de millas mayor es el cambio en el trazado de la ruta para evitar una

tormenta, evitar las zonas de mayor oleaje en ese instante o direcciones del oleaje no deseadas. Se desea ver si hay una relación directa entre la altura significativa de ola con la cantidad de millas extra a recorrer, o lo que es lo mismo, con la variación del trazado de la ruta respecto a la ruta de mínima distancia. En la figura 43 se puede observar la relación entre la altura significativa de ola con las millas extra recorridas.

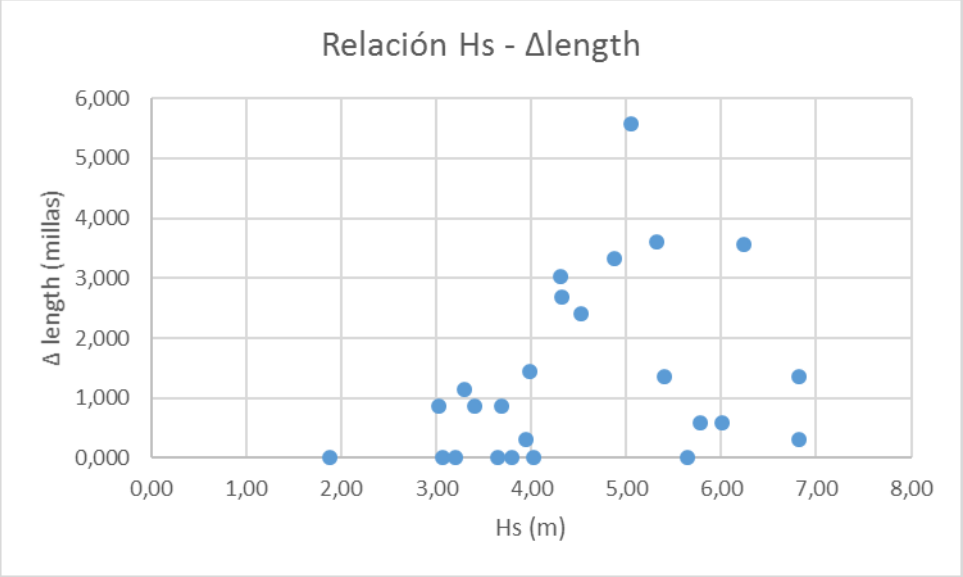
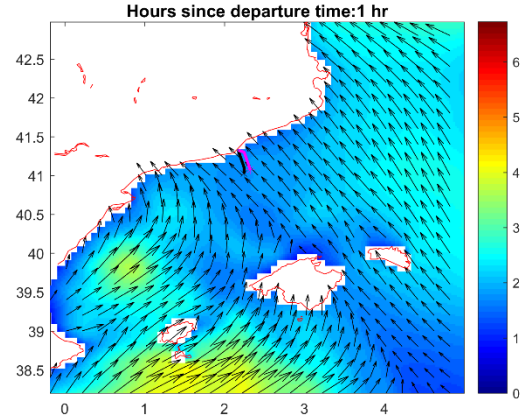


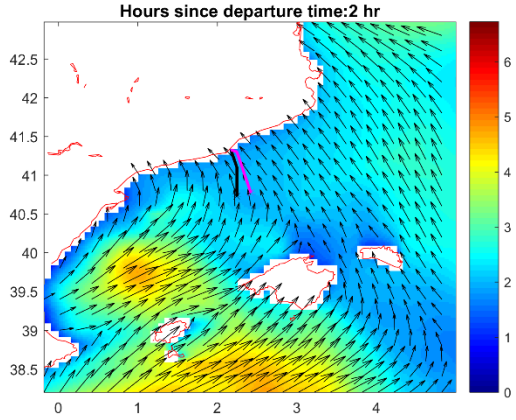
Figura 43: entre la ruta de distancia mínima considerando los efectos del oleaje y la ruta optimizada para las 25 simulaciones relacionando la altura significativa de ola con las millas extra recorridas por la ruta optimizada.

Semejante al caso anterior, tampoco se puede observar una relación directa entre la altura de ola significativa y el incremento en las millas recorridas, o lo que es lo mismo, en el cambio del trazado de la ruta.

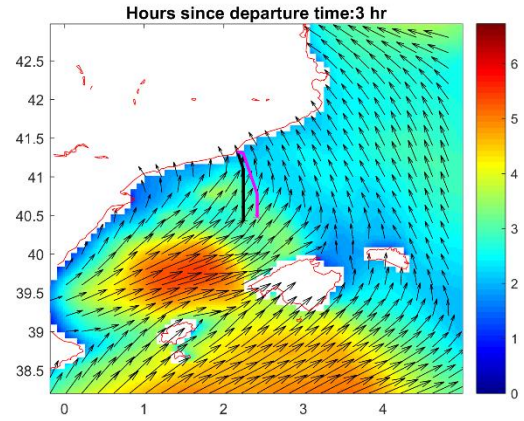
Como se ha podido observar en la figura 41 a veces el trazado de la ruta optimizada se ve muy afectado, no tanto por la altura significativa de ola, sino por el patrón espacial de la evolución de la tormenta. Otras veces el trazado viene muy condicionado por la dirección del oleaje más que la altura de este o del patrón espacial de la tormenta y su evolución, provocando que cambios en la dirección del oleaje en pocas horas puedan resultar en dos rutas óptimas de trazado muy diferentes una de la otra habiendo solo unas pocas horas de diferencia entre la salida de un navío y el otro. Un ejemplo de este suceso se puede observar en las figuras 44 y 45 donde se ha analizo la simulación del día 4/03/2017 considerando 2 horas de salida diferentes separadas solo por 2h de diferencia, a las 0:00h y a las 2:00h.



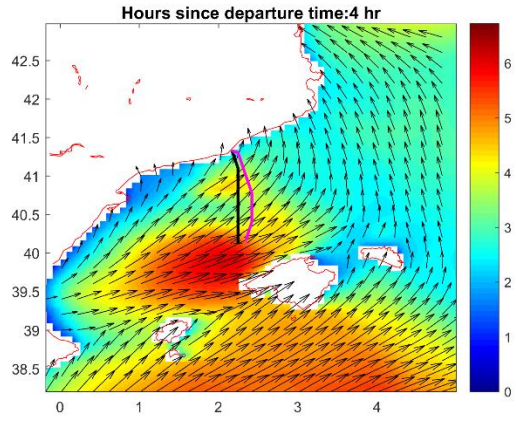
a) Avance de la ruta fija, la negra, y la ruta optimizada, la rosa, 1 h después de la salida.



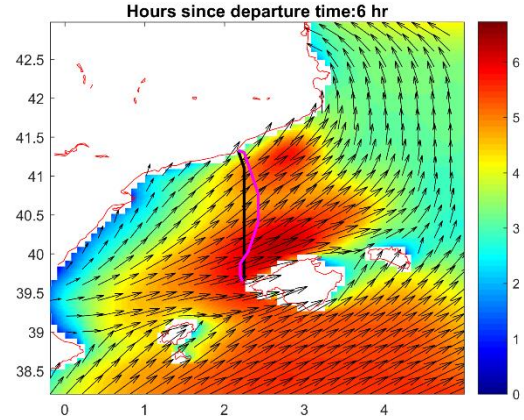
b) Avance de la ruta fija, la negra, y la ruta optimizada, la rosa, 2 h después de la salida.



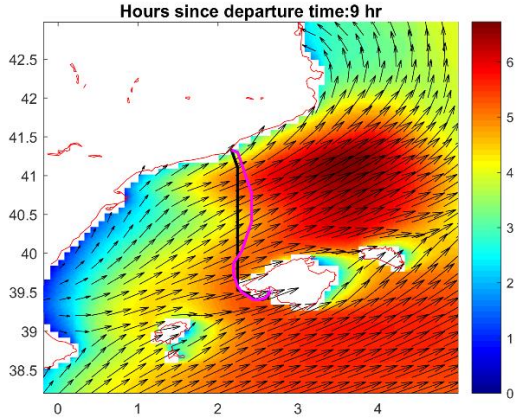
c) Avance de la ruta fija, la negra, y la ruta optimizada, la rosa, 3 h después de la salida.



d) Avance de la ruta fija, la negra, y la ruta optimizada, la rosa, 4 h después de la salida

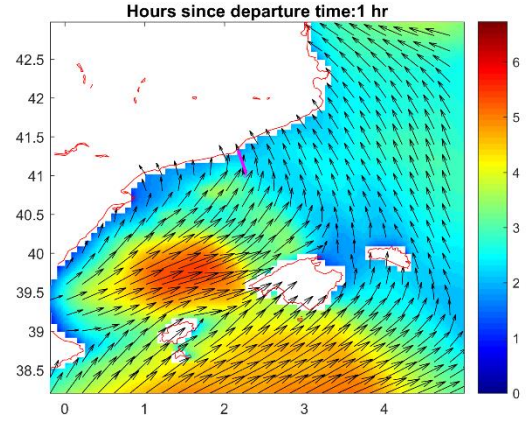


e) Avance de la ruta fija, la negra, y la ruta optimizada, la rosa, 6 h después de la salida.

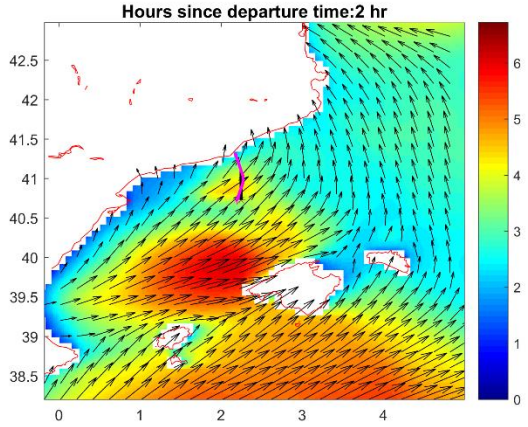


f) Llegada al puerto de destino de la ruta fija, la negra y la ruta optimizada, la rosa.

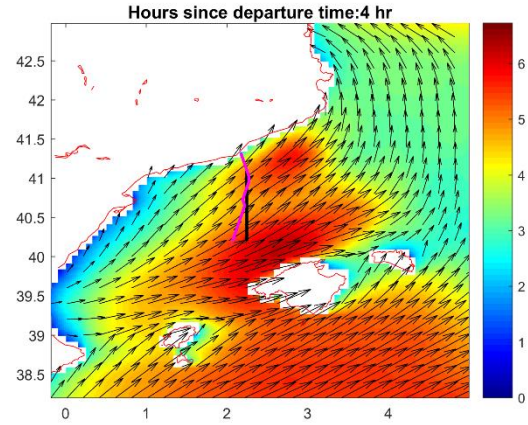
Figura 44: Evolución de las rutas fija y optimizada el día 04/03/2017 iniciando el viaje a las 0:00h.



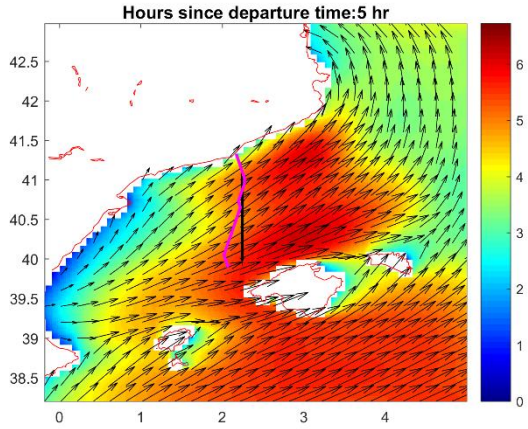
a) Avance de la ruta fija, la negra, y la ruta optimizada, la rosa, 1 h después de la salida.



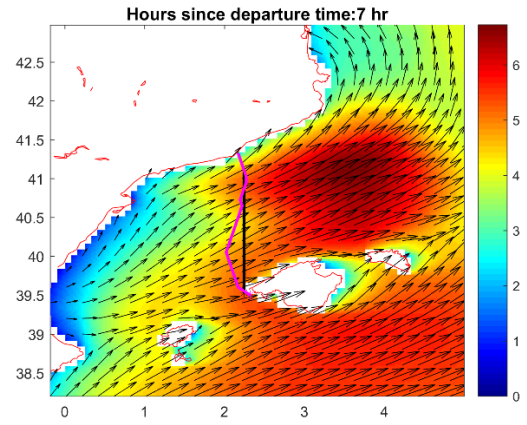
b) Avance de la ruta fija, la negra, y la ruta optimizada, la rosa, 2 h después de la salida.



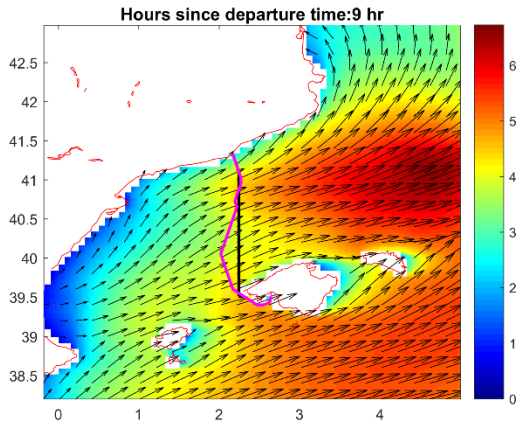
c) Avance de la ruta fija, la negra, y la ruta optimizada, la rosa, 4 h después de la salida.



d) Avance de la ruta fija, la negra, y la ruta optimizada, la rosa, 5 h después de la salida.



e) Avance de la ruta fija, la negra, y la ruta optimizada, la rosa, 7 h después de la salida.



f)) Llegada al puerto de destino de la ruta fija, la negra y la ruta optimizada, la rosa.

Figura 45: Evolución de las rutas fija y optimizada el día 04/03/2017 iniciando el viaje a las 2:00h.

En ambos casos la ruta optimizada ha intentado afrontar el oleaje de frente en las zonas de menor altura de ola significante para luego afrontar las zonas de mayor altura significante de ola en una dirección que produzca menos reducción de la velocidad debido al factor f relacionado con la dirección relativa entre el barco y la dirección de las olas. Se puede observar como en un período

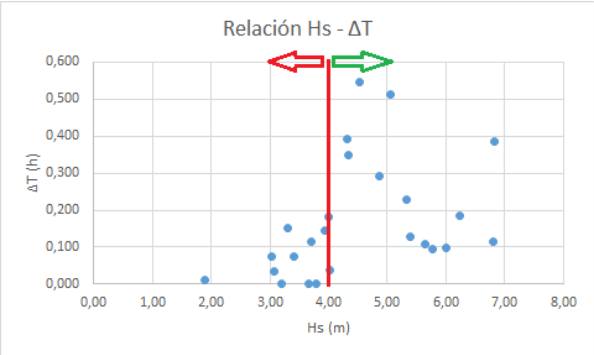
de 2 h la dirección del oleaje cerca del punto de salida había cambiado lo suficiente para obtener dos rutas finales que se desvían en diferentes direcciones respecto a la ruta de distancia mínima. Ambas rutas nuevas tienen un incremento similar, $\Delta length = 3{,}57\ millas$ para la que sale a las 00:00h y $\Delta length = 3{,}61\ millas$ para la que sale a las 02:00h, pero el trazado de ambas es radicalmente diferente

4.2.3. Impactos anuales

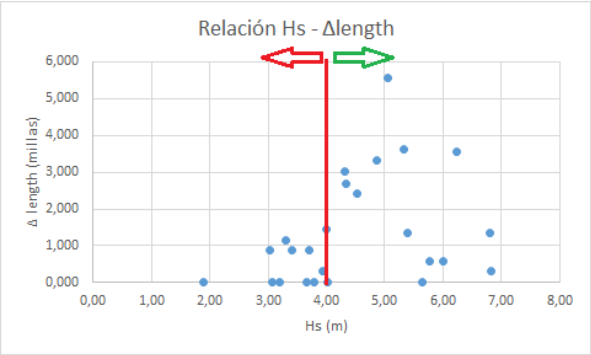
Se pretende realizar un estudio estadístico simple sobre el efecto que la aplicación de la optimización de rutas marítimas para viajes de corta duración puede tener a lo largo de un año a los viajes comerciales que actualmente se llevan a cabo entre el trayecto de Barcelona - Mallorca. Solo buscamos una aproximación para obtener un valor de referencia, no se pretende realizar un análisis exhaustivo con tal de determinar a la perfección el número de trayectos exactos.

Se ha deducido en el apartado anterior que el oleaje tiene un efecto en la ruta óptima mediante 3 factores: la altura significativa de ola, la dirección de oleaje y la evolución temporal del patrón espacial de la tormenta. De estos tres el más fácil de manejar es la altura significativa de ola, a pesar de que se ha comprobado que en algunos casos no afecta en mucho. Se ha decidido realizar el estudio estadístico en función solo de la altura de ola significativa, ya que el objetivo es obtener una aproximación sobre los viajes que podrían ver-se afectados y no un resultado exacto.

Como se ha descrito en el capítulo 2, Métodos, para el estudio estadístico es necesario definir una cota de altura de ola significativa a partir de la cual se considere que se produce una mejora en la ruta. Esta elección no es clara ya que como se ha explicado no es el único parámetro del que depende la ruta óptima. Como primera cota, se ha seleccionado los 4m, en la figura 46 se puede observar como a partir de los 4m se han incrementado considerablemente las reducciones de tiempo y el cambio del trazado respecto a la ruta de distancia mínima.



a) Límite de los 4 m en la relación altura significativa de ola con la reducción de tiempo.



b) Límite de los 4 m en la relación altura significativa de ola con el incremento de millas.

Figura 46: Visualización de la cota de 4m en la altura significativa de ola en los datos obtenidos en el presente trabajo.

Pero sabiendo que no siempre es la altura de ola significativa la que afecta más de los tres parámetros y que para alturas entre 3m y 4m se llega a reducciones de más de 10 min, que en un viaje de solo 8h pueden no ser despreciables, se ha considerado necesario definir otra cota, menor a 4m, y comparar ambos resultados. Esta otra cota se ha definido en los 3m de altura significativa de ola.

En la tabla 14 se resumen los datos obtenidos en el capítulo 3, Resultados.

Tabla 14: Cantidad de viajes afectados según la cota definida y porcentaje respecto a los viajes totales anuales realizados. (*para el año 2017 solo se dispone de datos de la primera mitad año)

Impacto anual				
Año	Nº viajes afectados (4m)	% del total de viajes al año (4m)	Nº viajes afectados (3m)	% del total de viajes al año (3m)
2013	78	6,23%	202	16,13%
2014	34	2,72%	110	8,79%
2015	58	4,63%	114	9,11%
2016	32	2,55%	96	7,66%
2017*	42	6,75%	82	13,18%

Se puede observar como en años tormentosos el programa tiene un gran impacto si los comparamos con años más calmados sin muchos temporales. Después de los análisis realizados, se puede suponer sin errar mucho que el número exacto de viajes afectados se encuentra entre los dos porcentajes obtenidos para cada año.

Para los dos años tormentosos entre los últimos, es decir, 2013 y 2017, el número de viajes afectados sería de entre 1.5/25 viajes realizados a 4/25 viajes realizados para el 2013 y de 1,5/25 viajes realizados a 3/25 viajes realizados para el 2017.

Hay que tener en cuenta que, en la actualidad, para esta ruta estudiada, Barcelona – Palma de Mallorca, se sigue la ruta de distancia mínima y en caso de día tormentoso o imprevistos que impiden ir según el horario establecido el capitán del navío da la orden de aumentar la potencia para llegar a la hora de llegada establecida. Esto implica un mayor consumo de fuel que se traduce en un aumento del coste en estos viajes que van fuera del horario esperado.

4.3. Conclusiones finales

El proyecto presentado en este trabajo es una comparación entre los algoritmos Dijkstra y A*y su implementación y uso en una herramienta de wheather routing desarrollada por la UPC, que se encuentra en sus fases iniciales. Actualmente solo dispone de la altura significativa de ola y la dirección relativa de las olas como inputs que modifican la velocidad del buque a la hora de realizar los cálculos de la optimización. Los datos obtenidos revelan que el algoritmo A* es más eficiente en comparación al algoritmo Dijkstra en cuanto a tiempo de computación. Pero hay que saber que en la optimización de rutas hay más parámetros que afectan a la ruta obtenida que no se han considerado en esta primera versión del SIMROUTE, como el efecto del viento y de las corrientes, las condiciones de seguridad a la hora de navegar en días con meteorología adversa para evitar la pérdida de cargamento o fallo de los sistemas del buque entre otros posibles efectos, etc. Futuros trabajos en este sistema incluyen la implementación de dynamic waves states , la inclusión de restricciones de seguridad debido a las condiciones del oleaje (p. ej. surfriding o rolling motions) en la metodología, la influencia de las corrientes o los vientos en la optimización del cálculo de rutas o la implementación de un algoritmo multi-criterio.

5. Bibliografía

Avgouleas, K. (2008). *Optimal ship routing.* Master's Tesis, Massachusetts Institute of Technology, Department of Mechanical Engineering, Cambridge, Maryland, US.

Bijlsma, S. J. (1975). *On minimal-timeship routing.* PhD thesis, Royal Netherlands Meteorological Institute, Delft University of Technology.

Bowditch, N. (2002). *The american practical navigator.* National Imagery and Mapping Agency, Bethesda, Maryland, USA.

Cai, Y., Wen, Y., & Wu, L. (2014). Ship Route Design for Avoiding Heavy Weather and Sea Conditions. *TransNav, the International Journal on Marine Navigation and Safety of Sea Transportation, Vol. 8*(No. 4), 551-556.

CAMINO MAS CORTO: ALGORITMO DE DIJKSTRA. (s.f.). Recuperado el 2016, de Algorithms and More: https://jariasf.wordpress.com/2012/03/19/camino-mas-corto-algoritmo-de-dijkstra/

Cavaleri, L. e. (2007). Wave modelling - The state of the art. *Progress in Oceanography, Vol. 75*(No. 4), 603-674.

Chen, H., Cardone, V., & Lacey, P. (1998). Use of operation support information technology to increase ship safety and efficiency. *SNAME transactions, Vol. 106*, 105-127.

Chu, P. C., Miller, S. E., & Hansen, J. A. (2015). Fuel-saving ship route using the Navy's ensemble meteorological and oceanic forecasts. *Journal of defense modeling and simulation, Vol. 12(1)*, 41-56.

Dijkstra, E. W. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik, Vol. 1*, 269-271.

Eranki, R. (2002). *Pathdinding using A* (A-Star).* Obtenido de Web.mit.edu: http://web.mit.edu/eranki/www/tutorials/search/

Grifoll, M. (2016). Ship Routing Applied at Short Sea Distances. En I. D. Politècnica (Ed.), *7th International Conference on Maritime Transport: technological, innovation and research: Maritime transport'16*, (págs. 240-245).

Grifoll, M., Navarro, J., Pallarès, E., Ràfols, L., Espino, M., & Palomares, A. (2016). *Oceanatmosphere-wave characterization of a wind jet (Ebro shelf NW Mediterranean Sea).* Discuss.

Gúzman Luna, J. A., Arango Sánchez, R. E., & Jiménez Pinzón, L. D. (2012). Búsqueda de la ruta óptima mediante los algoritmos: genéticos y dijkstra utilizando mapas de visibilidad. *Scientia et Technica*, 107-112.

Hagiwara, H. (1982). *Weather routing of (sail-assisted) motorvessel.* PhD thesis, University of Tokyo, Tokyo, Japón.

Hinnenthal, J. (2008). *Robust Pareto optimum routing of ships utilizing deterministic and ensemble weather forecasts.* PhD thesis, Technischen Universität Berlin, Berlin, Alemania.

Hu, Q., Cai, F., Yang, C., & Shi, C. J. (2014). An algorithm for Interpolating Ship Motion Vectors. *TransNav, the International Journal on Marine Navigation and Safety of Sea Transportation, Vol. 8*(No. 1), 35-40.

Kirk, J. (21 de Mayo de 2007). *Dijkstra's Shortest Path Algorithm*. Recuperado el 2016, de Mathworks: http://es.mathworks.com/matlabcentral/fileexchange/12850-dijkstra-s-shortest-path-algorithm

Klompstra, M. B., Olsde, G. J., & Van Brunschot, P. (1992). The isopone method in optimal control. *Dynamics and Control, Vol. 2 (3)*, 281-301.

Larsson, E., & Simonsen, M. H. (2014). *DIRECT weather routing.* Master's thesis, Chalmers University of Technology, Department of Shipping and Marine Technology, Gothenburg, Suecia.

Maki, A., Akimoto, Y., Nagata, Y., Kobayashi, S., Kobayashi, E., Shiotani, S., . . . Umeda, N. (2011). A new weather-routing system with real-coded genetic algorithm. *Journal of Marine Science and TEchnology, Vol. 16*, 311-322.

Mannarini, G., Coppini, G., Oddo, P., & Pinardi, N. (2013). A Prototype of Ship Routing Decision Support System for an Operational Oceanographic Service. *TransNav, the International Journal on Marine Navigation and Safety of Sea Transportation, Vol. 7*(No. 1), 53-59.

Mapa en vivo de situación y tráfico marítimo de barcos con AIS | AIS Marine Traffic. (s.f.). Recuperado el 2017, de Marinetraffic: https://www.marinetraffic.com/es/ais/home/centerx:3.0/centery:40.0/zoom:7

Marie, S., & Courteille, E. (2099). Multi-Objective Optimization of Motor Vessel Route. *International Journal on Marine Navigation and Safety of Sea Transportation, Vol. 3*(No. 2), 133-141.

Martínez de Osés, F. X., & Castells, M. (2008). Heavy Weather in European Short Sea Shipping: Its Influences on Selected Routes. *The Journal of Navigation, Vol. 61*, 165-176.

Mechatronics, U. (24 de Junio de 2013). *Dijkstra's Algorithm vs. A* Search vs. Concurrent Dijkstra's Algorithm*. Recuperado el 2016, de YouTube: https://www.youtube.com/watch?v=cSxnOm5aceA.

Montes, A. A. (2005). *Network shortest path application for optimum track ship routing.* PhD thesis, Monterey Naval Postgraduate School.

Prediccion de oleaje, nivel del mar; Boyas y mareografos | puertos.es. (s.f.). Recuperado el 2017, de Puertos.es: http://www.puertos.es/es-es/oceanografia/Paginas/portus.aspx

Premakur, P. (01 de Septiembre de 2016). *Dijkstra's Shortest Path Algorithm*. Recuperado el 2016, de Mathworks: http://es.mathworks.com/matlabcentral/fileexchange/12850-dijkstra-s-shortest-path-algorithm

Puthuparampil, M. (s.f.). *Report Dijkstra's Algorithm.* Report.

Reddy, H. (13 Diciembre 2013). *Path Finding - Dijkstra's and A* Algorithm's.*

Salas, A. H. (1 Octubre 2008). *Acerca del Algoritmo de Dijkstra.* National University of Colombia, Departamento de Matemáticas y Estadística (Manizales).

Shao, W. (2013). *Development of an intelligent tool for energy efficient and low environment impact shipping.* PhD thesis, University of Strathclyde, UK.

Shao, W., Zhou, P., & Thong, S. K. (2012). Development of a novel forward dynamic programming method for weather routing. *Journal of Marine Science and Technology, Vol. 17*, 239-251.

Simonsen, M. H., Larsson, E., Mao, W., & Ringsberg, J. W. (31 mayo - 5 Junio, 2015). State-of-the-Art within ship weather routing. *Proceedings of the ASME 2015 34th International COnference on Ocean, Offshore and Artic Engineering.* St. John's, Newfoundland, Canada.

TdsStaticCatalog. (s.f.). Recuperado el 2017, de Opendap.puertos.es: http://opendap.puertos.es/thredds/catalog.html

6. Anexos

6.1. Código Dijkstra

El código del algoritmo Dijkstra está compuesto por el script principal llamado *coditfgDanimacio.m* y por dos funciones necesarias para su funcionamiento: *dijkstra1.m*, la cual busca en las 8 direcciones posibles si el movimiento es posible dado un nodo y *dibujar.m*, que permite la representación de la solución mediante un código numérico de colores.

6.1.1. coditfgDanimacio.m

```
%%TFG: Algoritmo Dijkstra

    %Lluis Martorell Pons%
clear all; close all; %clc
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Introducir nº filas y columnas del mapa
Filas=34;
Columnas=34;
MAP = zeros(Filas,Columnas);
%Introducir obstaculos:
MAP (14:21,27)=ones(8,1);
MAP (14,20:27)=ones(1,8);
% Introducir puntos origen y destino:
PO= [16,18];     %Punto origen (PO).
PF = [16,29];    %Punto final (PF).
%************************************************************************%
%%Inicializamos matriz de costes%% {Consideramos coste infinito como 1000}
Coste = 1000*ones(Filas,Columnas);
Coste(PO(1),PO(2))=0;
%************************************************************************%
%%Creamos los conjuntos que estudiaremos%%
NC=[0,0,0];
NA=[PO,Coste(PO(1),PO(2))];
Predecesores=[0,0,PO];
N_prev=[];
Camino_min=[];
%************************************************************************%
%%Preparamos los elementos para la representación de los resultados%%
%Definimos los colormaps:
    %gris (0's) -> Nodos no estudiados
    %negro (1's) -> Obstaculos
    %rojo (2's) -> Nodos cerrados
    %verde (3's) -> Nodos abiertos
    %azul (4's) -> Camino mínimo
    %amarillo (5's) -> Nodos origen y destino
colores0=[0.5,0.5,0.5;0,0,0;1,1,0];                          %'inicial' Caso
inicial, solo MAP i PO-PF.
colores1=[0.5,0.5,0.5;0,0,0;0,1,0;1,1,0];                    %'iteracion1'
durante la 1ª iteracion-> MAP,NA,PO-PF
```

```
colores2=[0.5,0.5,0.5;0,0,0;0,1,0;1,0,0;1,1,0];              %'iteracion'
durante la 2ª iteracion (y posteriores)-> MAP,NA,NC,PO-PF
coloresfinal=[0.5,0.5,0.5;0,0,0;0,1,0;1,0,0;0,0,1;1,1,0];   %'acabado' Para la
solucion final-> MAP,NA,NC,Camino_min,PO-PF
%Preparamos figura y ejes:
figure (1);
X=[1:Columnas+1];
Y=flip([1:Filas+1]);
%Dibujamos el momento de inicio:
Dibujo=dibujar(Filas,Columnas,MAP,NA,NC,Camino_min,PO,PF,'inicial');
colormap (figure (1),colores0);
pcolor(X,Y,Dibujo);
set(gcf,'PaperPositionMode','auto')
print ('-dpng','-opengl','-r300',num2str(0))
%Iniciamos el contador para los nombres durante las iteraciones
i=1;
%************************************************************************%
tic;
while (isempty (NA) == 0)
    NA=sortrows(NA,3);      %Ordenamos por menor coste.
    %%Comprobamos si hemos llegado al destino deseado%%
    if NA(1,1)==PF(1) & NA(1,2)==PF(2)
        NC=[NC;NA(1,:)];    %Metemos el destino en nodos cerrados.
        %Actualizamos predecesores%
        if isempty(find(Predecesores(:,3)==PF(1)&Predecesores(:,4)==PF(2)))==1
                Predecesores=[Predecesores;N_prev(1),N_prev(2),PF(1),PF(2)];
            else

Predecesores(find(Predecesores(:,3)==PF(1)&Predecesores(:,4)==PF(2)),1:2) =
[N_prev(1),N_prev(2)];    %Sí esta -> Cambiamos el origen para llegar a N_n.
            end
        NA(1,:)=[];         %Eliminamos el nodo destino de los nodos abiertos.
{{Y paramos el Dijkstra}}
        break
    end
    N_sel=NA(1,1:2);
    %Llamamos la función Astar1 para cada una de las direcciones de movimiento
possibles:
    [Coste,NA,Predecesores,N_prev]=dijkstra1
(MAP,Filas,Columnas,PF,Coste,NA,NC,Predecesores,N_prev,N_sel,'n');
    [Coste,NA,Predecesores,N_prev]=dijkstra1
(MAP,Filas,Columnas,PF,Coste,NA,NC,Predecesores,N_prev,N_sel,'ne');
    [Coste,NA,Predecesores,N_prev]=dijkstra1
(MAP,Filas,Columnas,PF,Coste,NA,NC,Predecesores,N_prev,N_sel,'e');
    [Coste,NA,Predecesores,N_prev]=dijkstra1
(MAP,Filas,Columnas,PF,Coste,NA,NC,Predecesores,N_prev,N_sel,'se');
    [Coste,NA,Predecesores,N_prev]=dijkstra1
(MAP,Filas,Columnas,PF,Coste,NA,NC,Predecesores,N_prev,N_sel,'s');
    [Coste,NA,Predecesores,N_prev]=dijkstra1
(MAP,Filas,Columnas,PF,Coste,NA,NC,Predecesores,N_prev,N_sel,'sw');
    [Coste,NA,Predecesores,N_prev]=dijkstra1
(MAP,Filas,Columnas,PF,Coste,NA,NC,Predecesores,N_prev,N_sel,'w');
    [Coste,NA,Predecesores,N_prev]=dijkstra1
(MAP,Filas,Columnas,PF,Coste,NA,NC,Predecesores,N_prev,N_sel,'nw');
    %Actualizamos NC y NA:%
    NC=[NC;NA(1,:)];     %Ponemos el nodo en nodos cerrados.
    NA(1,:)=[];          %Eliminamos el nodo de la frontera.
    %Representamos la iteracion:
    Dibujo=dibujar(Filas,Columnas,MAP,NA,NC,Camino_min,PO,PF,'iteracion');
    if i==1
```

```
        colormap (figure (1),colores1);
    else colormap (figure (1),colores2);
    end
    pcolor(X,Y,Dibujo);
    set(gcf,'PaperPositionMode','auto')
    print ('-dpng','-opengl','-r300',num2str(i))
    %Actualizamos el nº de iteracion:
    i=i+1;
end
toc;
%%Obtenemos el camino recorrido%%
N_k=N_prev;              %Nodo buscado en Predecesores (coordenadas destinos).
Busqueda=[PF;N_k];         %Lista de los nodos que componen el camíno óptimo a
PF.
while ne(N_k(1)-PO(1),0)+ne(N_k(2)-PO(2),0)==0==0
N_k=Predecesores(find(Predecesores(:,3)==N_k(1)&Predecesores(:,4)==N_k(2)),1:2)
;
Busqueda=[Busqueda;N_k];
end
Camino_min=flip(Busqueda);
%Representación gráfica de la solución final%
Dibujo=dibujar(Filas,Columnas,MAP,NA,NC,Camino_min,PO,PF,'acabado');
colormap (figure (1),coloresfinal);
pcolor(X,Y,Dibujo)
set(gcf,'PaperPositionMode','auto')
print ('-dpng','-opengl','-r300','finalD')
```

6.1.2. dijkstra1.m

```
function [Coste,NA,Predecesores,N_prev] =
dijkstra1(MAP,Filas,Columnas,PF,Coste,NA,NC,Predecesores,N_prev,N_sel,
direccion )
%dijkstra1 - Calcula una iteracion del algoritmo Dijkstra
%   MAP -> Mapa binario que define los obstaculos.
%   Filas -> Filas de MAP.
%   Columnas -> Columnas de MAP.
%   Coste -> Matriz de costes {inicialmente inf}.
%   NA -> Matriz nodos abiertos.
%   NC -> Matriz nodos cerrados.
%   Predecesores -> Matriz caminos realizados.
%   N_prev -> Posición para encontrar el camino optimo al final.
%   N_sel -> Nodo del que estudiamos los vecinos.
%   direccion -> 'n', 'ne', 'e', 'se', 's', 'sw', 'w', 'nw'.

% Buscamos en que direccion nos encontramos y aplicamos las condiciones a
imponer en ella:
if direccion == 'n'
    N=N_sel + [-1,0];
    coste=1;
    %Comprobamos que no salimos de MAP y que el coste sea menor.
    if N(1)>0 & Coste(N_sel(1),N_sel(2))+1 < Coste(N(1),N(2))
        movimiento = true;
    else movimiento = false;
    end
elseif direccion == 'ne'
    N=N_sel + [-1,+1];
    coste=sqrt(2);
    %Comprobamos que no salimos de MAP y que el coste sea menor.
```

```
    if N(1)>0 & N(2)<Columnas+1 & Coste(N_sel(1),N_sel(2))+1.4 <
Coste(N(1),N(2))
        movimiento = true;
    else movimiento = false;
    end
elseif direccion == 'e'
    N=N_sel + [0,+1];
    coste=1;
    %Comprobamos que no salimos de MAP y que el coste sea menor.
    if N(2)<Columnas+1 & Coste(N_sel(1),N_sel(2))+1 < Coste(N(1),N(2))
        movimiento = true;
    else movimiento = false;
    end
elseif direccion == 'se'
    N=N_sel + [+1,+1];
    coste=sqrt(2);
    %Comprobamos que no salimos de MAP y que el coste sea menor.
    if N(1)<Filas+1 & N(2)<Columnas+1 & Coste(N_sel(1),N_sel(2))+1.4 <
Coste(N(1),N(2))
        movimiento = true;
    else movimiento = false;
    end
elseif direccion == 's'
    N=N_sel + [+1,0];
    coste=1;
    %Comprobamos que no salimos de MAP y que el coste sea menor.
    if N(1)<Filas+1 & Coste(N_sel(1),N_sel(2))+1 < Coste(N(1),N(2))
        movimiento = true;
    else movimiento = false;
    end
elseif direccion == 'sw'
    N=N_sel + [+1,-1];
    coste=sqrt(2);
    %Comprobamos que no salimos de MAP y que el coste sea menor.
    if N(1)<Filas+1 & N(2)>0 & Coste(N_sel(1),N_sel(2))+1.4 < Coste(N(1),N(2))
        movimiento = true;
    else movimiento = false;
    end
elseif direccion == 'w'
    N=N_sel + [0,-1];
    coste=1;
    %Comprobamos que no salimos de MAP y que el coste sea menor.
    if N(2)>0 & Coste(N_sel(1),N_sel(2))+1 < Coste(N(1),N(2))
        movimiento = true;
    else movimiento = false;
    end
elseif direccion == 'nw'
    N=N_sel + [-1,-1];
    coste=sqrt(2);
    %Comprobamos que no salimos de MAP y que el coste sea menor.
    if N(1)>0 & N(2)>0 & Coste(N_sel(1),N_sel(2))+1.4 < Coste(N(1),N(2))
        movimiento = true;
    else movimiento = false;
    end
else movimiento = false;
end
% Si se cumplen las condiciones segun "direccion" y no se encuentra el nodo en
NC o hay obstaculos en MAP:
if movimiento == true & isempty(find(NC(:,1)==N(1) & NC(:,2)==N(2))) &
MAP(N(1),N(2))==0
```

```
    %Actualizamos coste:
    Coste(N(1),N(2))=Coste(N_sel(1),N_sel(2))+coste;
    %Actualizamos NA%
    if isempty(find(NA(:,1)==N(1)&NA(:,2)==N(2)))==1    %Buscamos si N esta en
NA.
        NA=[NA;N,Coste(N(1),N(2))];
    else
        NA(find(NA(:,1)==N(1)&NA(:,2)==N(2)),3) = Coste(N(1),N(2));
    end
    %Actualizamos Predecesores%
    if isempty(find(Predecesores(:,3)==N(1)&Predecesores(:,4)==N(2)))==1
%Buscamos si N esta en Predecesores como un destino.
        Predecesores=[Predecesores;N_sel(1),N_sel(2),N(1),N(2)];
%Introducimos N_sel como origen i N como destino.
    else
        Predecesores(find(Predecesores(:,3)==N(1)&Predecesores(:,4)==N(2)),1:2)
= N_sel;    %Cambiamos el nodo para llegar a N.
    end
    %Actualizamos N_prev:%
    if N(1)==PF(1) & N(2)==PF(2)
        N_prev=N_sel;
    end
end
end
```

6.1.3. dibujar.m

```
function [ Dibujo ] = dibujar( Filas,Columnas,MAP,NA,NC,Camino_min,PO,PF,etapa)
%dibujar Función que prepara la matriz ha utilizar con pcolor para
%representar los resultados de aplicar los algoritmos Dijkstra y A*
%   Filas -> Filas de MAP.
%   Columnas -> Columnas de MAP.
%   MAP -> Mapa binario que define los obstaculos.
%   NA -> Matriz nodos abiertos.
%   NC -> Matriz nodos cerrados.
%   Camino_min -> Lista de vectores que indican las coordenadas de los puntos
por los que pasa el camino optimo.
%   PO -> Vector de posicion inicial.
%   PF -> Vector de posicion final.
%   etapa -> String que indicará que valores dar a la matriz para poder usar
luego el colormap adecuado.
                % 'inicial' -> Sin inicializar el algoritmo
                % 'iteracion1' -> Después de la primera iteración
                % 'iteracion' -> 2 y siguientes iteraciones
                % 'acabado' -> Caso final con Camino_min.

%Definimos la matriz con todos los datos a representar:
Dibujo=[zeros(Filas+1,Columnas+1)];
%Indicamos los obstaculos
for F=1:Filas
    for C=1:Columnas
        if MAP(F,C)==1
            Dibujo(F,C)=1;
        end
    end
end
%Indicamos nodos abiertos
if strcmp(etapa,'inicial')
else
```

```
    contador = size(NA);
    for F=1:contador(1)
        Dibujo(NA(F,1),NA(F,2))=2;
    end
end
%Indicamos nodos cerrados
if strcmp(etapa,'inicial')
elseif strcmp(etapa,'iteracion1')
else
    contador = size(NC);
    for F=2:contador(1)
        Dibujo(NC(F,1),NC(F,2))=3;
    end
end
%Indicamos camino minimo:
if strcmp(etapa,'acabado')
    contador = size(Camino_min);
    for F=1:contador(1)
        Dibujo(Camino_min(F,1),Camino_min(F,2))=4;
    end
else
end
%Indicamos PO y PF:
if strcmp(etapa,'inicial')
    Dibujo(PO(1),PO(2))=2;
    Dibujo(PF(1),PF(2))=2;
elseif strcmp(etapa,'iteracion1')
    Dibujo(PO(1),PO(2))=3;
    Dibujo(PF(1),PF(2))=3;
elseif strcmp(etapa,'iteracion')
    Dibujo(PO(1),PO(2))=4;
    Dibujo(PF(1),PF(2))=4;
else
    Dibujo(PO(1),PO(2))=5;
    Dibujo(PF(1),PF(2))=5;
end
```

6.2. Código A*

El código del algoritmo A* está compuesto por el script principal llamado *coditfgAanimacio.m* y por tres funciones necesarias para su funcionamiento: *Astar1.m*, la cual busca en las 8 direcciones posibles si el movimiento es posible dado un nodo, *costeheuristica.m* que calcula el coste de la heurística dados el nodo final y el nodo origen y *dibujar.m*, la misma función utilizada en el código del Dijkstra.

6.2.1. coditfgAanimacio.m

```
%%TFG: Algoritme A*

    %Lluis Martorell Pons%
clear all; %close all; %clc
%% %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%Introducir nº filas y columnas del mapa
Filas=34;
Columnas=34;
MAP = zeros(Filas,Columnas);
```

```
%Introducir obstaculos:
MAP (14:18,20)=ones(5,1);
% Introducir puntos origen y destino:
PO= [16,18];      %Punto origen (PO).
PF = [16,22];     %Punto final (PF).
%************************************************************************%
%%Inicializamos matriz de costes%% {Consideramos coste infinito como 1000}
CosteO = 1000*ones(Filas,Columnas);      %Coste distancia al origen.
CosteO(PO(1),PO(2))=0;
CosteF = 1000*ones(Filas,Columnas);      %Coste distancia al destino.
CosteF(PO(1),PO(2))=sqrt((PF(1)-PO(1))^2+(PF(2)-PO(2))^2);
Costetotal = CosteO+CosteF;              %Suma de ambos costes anteriores.
%************************************************************************%
%%Creamos los conjuntos que estudiaremos%%
NC=[0,0,0,0,0];                  %Nodos cerrados == Los nodes que el algoritmo
ya ha optimizado.
NA=[PO,CosteO(PO(1),PO(2)),CosteF(PO(1),PO(2)),Costetotal(PO(1),PO(2))]; %Nodos
abiertos == Frontera de nodos estudiados.
Predecesores=[0,0,PO];       %Lista de nodo origen --> nodo destino para cada
movimiento.{ponemos el caso 0 ya que sino Matlab no funciona en la primera
iteracion}
N_prev=[];                   %Nodo desde el cual llegamos al origen.
Camino_min=[];               %Matriz con los nodos (por filas) que componen el
camino mínimo.
%************************************************************************%
%%Preparamos los elementos para la representación de los resultados%%
%Definimos los colormaps:
    %gris (0's) -> Nodos no estudiados
    %negro (1's) -> Obstaculos
    %rojo (2's) -> Nodos cerrados
    %verde (3's) -> Nodos abiertos
    %azul (4's) -> Camino mínimo
    %amarillo (5's) -> Nodos origen y destino
colores0=[0.5,0.5,0.5;0,0,0;1,1,0];                          %'inicial' Caso
inicial, solo MAP i PO-PF.
colores1=[0.5,0.5,0.5;0,0,0;0,1,0;1,1,0];                    %'iteracion1'
durante la 1ª iteracion-> MAP,NA,PO-PF
colores2=[0.5,0.5,0.5;0,0,0;0,1,0;1,0,0;1,1,0];              %'iteracion'
durante la 2ª iteracion (y posteriores)-> MAP,NA,NC,PO-PF
coloresfinal=[0.5,0.5,0.5;0,0,0;0,1,0;1,0,0;0,0,1;1,1,0];    %'acabado' Para la
solucion final-> MAP,NA,NC,Camino_min,PO-PF
%Preparamos figura y ejes:
figure (2);
X=[1:Columnas+1];
Y=flip([1:Filas+1]);
%Dibujamos el momento de inicio:
Dibujo=dibujar(Filas,Columnas,MAP,NA,NC,Camino_min,PO,PF,'inicial');
colormap (figure (2),colores0);
pcolor(X,Y,Dibujo);
set(gcf,'PaperPositionMode','auto')
print ('-dpng','-opengl','-r300',num2str(1000))
%Iniciamos el contador para los nombres durante las iteraciones
i=1001;
%************************************************************************%
tic;
while (isempty (NA) == 0)
    NA=sortrows(NA,5);       %Ordenamos por menor costetotal.
    %%Comprobamos si hemos llegado al destino deseado%%
    if NA(1,1)==PF(1) & NA(1,2)==PF(2)
        NC=[NC;NA(1,:)];     %Metemos el destino en nodos cerrados.
```

```
        %Actualizamos predecesores%
        if isempty(find(Predecesores(:,3)==PF(1)&Predecesores(:,4)==PF(2)))==1
                Predecesores=[Predecesores;N_prev(1),N_prev(2),PF(1),PF(2)];
%El origen será el nodo seleccionado en la iteracion anterior.
            else

Predecesores(find(Predecesores(:,3)==PF(1)&Predecesores(:,4)==PF(2)),1:2) =
[N_prev(1),N_prev(2)];    %Sí esta -> Cambiamos el origen para llegar a N_n.
            end
        NA(1,:)=[];          %Eliminamos el nodo destino de los nodos abiertos.
{{Y paramos el Dijkstra}}
        break
    else
    N_sel=NA(1,1:2);
    %Llamamos la función Astar1 para cada una de las direcciones de movimiento
possibles:
    [CosteO,CosteF,Costetotal,NA,Predecesores,N_prev]=Astar1
(MAP,Filas,Columnas,PF,CosteO,CosteF,Costetotal,NA,NC,Predecesores,N_prev,N_sel
,'n');
    [CosteO,CosteF,Costetotal,NA,Predecesores,N_prev]=Astar1
(MAP,Filas,Columnas,PF,CosteO,CosteF,Costetotal,NA,NC,Predecesores,N_prev,N_sel
,'ne');
    [CosteO,CosteF,Costetotal,NA,Predecesores,N_prev]=Astar1
(MAP,Filas,Columnas,PF,CosteO,CosteF,Costetotal,NA,NC,Predecesores,N_prev,N_sel
,'e');
    [CosteO,CosteF,Costetotal,NA,Predecesores,N_prev]=Astar1
(MAP,Filas,Columnas,PF,CosteO,CosteF,Costetotal,NA,NC,Predecesores,N_prev,N_sel
,'se');
    [CosteO,CosteF,Costetotal,NA,Predecesores,N_prev]=Astar1
(MAP,Filas,Columnas,PF,CosteO,CosteF,Costetotal,NA,NC,Predecesores,N_prev,N_sel
,'s');
    [CosteO,CosteF,Costetotal,NA,Predecesores,N_prev]=Astar1
(MAP,Filas,Columnas,PF,CosteO,CosteF,Costetotal,NA,NC,Predecesores,N_prev,N_sel
,'sw');
    [CosteO,CosteF,Costetotal,NA,Predecesores,N_prev]=Astar1
(MAP,Filas,Columnas,PF,CosteO,CosteF,Costetotal,NA,NC,Predecesores,N_prev,N_sel
,'w');
    [CosteO,CosteF,Costetotal,NA,Predecesores,N_prev]=Astar1
(MAP,Filas,Columnas,PF,CosteO,CosteF,Costetotal,NA,NC,Predecesores,N_prev,N_sel
,'nw');
    %Actualizamos NC y NA:%
    NC=[NC;NA(1,:)];     %Ponemos el nodo en nodos cerrados.
    NA(1,:)=[];          %Eliminamos el nodo de la frontera.
    %Representamos la iteracion:
    Dibujo=dibujar(Filas,Columnas,MAP,NA,NC,Camino_min,PO,PF,'iteracion');
    if i==1001
        colormap (figure (2),colores1);
    else colormap (figure (2),colores2);
    end
    pcolor(X,Y,Dibujo);
    set(gcf,'PaperPositionMode','auto')
    print ('-dpng','-opengl','-r300',num2str(i))
    %Actualizamos el nº de iteracion:
    i=i+1;
    end
end
temps=toc;         %Obtenemos cuanto ha tardado.
%%Obtenemos el camino recorrido%%
N_k=N_prev;              %Nodo buscado en Predecesores (coordenadas destinos).
```

```
Busqueda=[PF;N_k];          %Lista de los nodos que componen el camíno óptimo a
PF.
while ne(N_k(1)-PO(1),0)+ne(N_k(2)-PO(2),0)==0==0
N_k=Predecesores(find(Predecesores(:,3)==N_k(1)&Predecesores(:,4)==N_k(2)),1:2)
;
Busqueda=[Busqueda;N_k];
end
Camino_min=flip(Busqueda);
%Representación gráfica de la solución final%
Dibujo=dibujar(Filas,Columnas,MAP,NA,NC,Camino_min,PO,PF,'acabado');
colormap (figure (2),coloresfinal);
pcolor(X,Y,Dibujo)
set(gcf,'PaperPositionMode','auto')
print ('-dpng','-opengl','-r300','finalA')
```

6.2.2. Astar1.m

```
function [ CosteO, CosteF, Costetotal,NA,Predecesores,N_prev ] = Astar1( MAP,
Filas, Columnas, PF, CosteO, CosteF, Costetotal, NA, NC, Predecesores, N_prev,
N_sel, direccion )
%Astar1 - Calcula una iteracion del algoritmo A*
%   MAP -> Mapa binario que define los obstaculos.
%   Filas -> Filas de MAP.
%   Columnas -> Columnas de MAP.
%   Coste -> Matriz de costes {inicialmente inf}.
%   NA -> Matriz nodos abiertos.
%   NC -> Matriz nodos cerrados.
%   Predecesores -> Matriz de caminos realizados (parejas de nodos).
%   N_prev -> Posición para encontrar el camino óptimo al final.
%   N_sel -> Nodo del que estudiamos los vecinos.
%   direccion -> 'n', 'ne', 'e', 'se', 's', 'sw', 'w', 'nw'.

% Buscamos en que direccion nos encontramos y aplicamos las condiciones a
imponer en ella:
if direccion == 'n'
    N=N_sel + [-1,0];
    coste=1;
    %Comprobamos que no salimos de MAP y que el coste sea menor.
    if N(1)>0 & CosteO(N_sel(1),N_sel(2))+1+sqrt((PF(1)-N(1))^2+(PF(2)-
N(2))^2)< Costetotal(N(1),N(2))
        movimiento = true;
    else movimiento = false;
    end
elseif direccion == 'ne'
    N=N_sel + [-1,+1];
    coste=sqrt(2);
    %Comprobamos que no salimos de MAP y que el coste sea menor.
    if N(1)>0 & N(2)<Columnas+1 & CosteO(N_sel(1),N_sel(2))+1.4+sqrt((PF(1)-
N(1))^2+(PF(2)-N(2))^2)< Costetotal(N(1),N(2))
        movimiento = true;
    else movimiento = false;
    end
elseif direccion == 'e'
    N=N_sel + [0,+1];
    coste=1;
    %Comprobamos que no salimos de MAP y que el coste sea menor.
    if N(2)<Columnas+1 & CosteO(N_sel(1),N_sel(2))+1+sqrt((PF(1)-
N(1))^2+(PF(2)-N(2))^2)< Costetotal(N(1),N(2))
```

```
        movimiento = true;
    else movimiento = false;
    end
elseif direccion == 'se'
    N=N_sel + [+1,+1];
    coste=sqrt(2);
    %Comprobamos que no salimos de MAP y que el coste sea menor.
    if N(1)<Filas+1 & N(2)<Columnas+1 &
CosteO(N_sel(1),N_sel(2))+1.4+sqrt((PF(1)-N(1))^2+(PF(2)-N(2))^2)<
Costetotal(N(1),N(2))
        movimiento = true;
    else movimiento = false;
    end
elseif direccion == 's'
    N=N_sel + [+1,0];
    coste=1;
    %Comprobamos que no salimos de MAP y que el coste sea menor.
    if N(1)<Filas+1 & CosteO(N_sel(1),N_sel(2))+1+sqrt((PF(1)-N(1))^2+(PF(2)-
N(2))^2)< Costetotal(N(1),N(2))
        movimiento = true;
    else movimiento = false;
    end
elseif direccion == 'sw'
    N=N_sel + [+1,-1];
    coste=sqrt(2);
    %Comprobamos que no salimos de MAP y que el coste sea menor.
    if N(1)<Filas+1 & N(2)>0 & CosteO(N_sel(1),N_sel(2))+1.4+sqrt((PF(1)-
N(1))^2+(PF(2)-N(2))^2)< Costetotal(N(1),N(2))
        movimiento = true;
    else movimiento = false;
    end
elseif direccion == 'w'
    N=N_sel + [0,-1];
    coste=1;
    %Comprobamos que no salimos de MAP y que el coste sea menor.
    if N(2)>0 & CosteO(N_sel(1),N_sel(2))+1+sqrt((PF(1)-N(1))^2+(PF(2)-
N(2))^2)< Costetotal(N(1),N(2))
        movimiento = true;
    else movimiento = false;
    end
elseif direccion == 'nw'
    N=N_sel + [-1,-1];
    coste=sqrt(2);
    %Comprobamos que no salimos de MAP y que el coste sea menor.
    if N(1)>0 & N(2)>0 & CosteO(N_sel(1),N_sel(2))+1.4+sqrt((PF(1)-
N(1))^2+(PF(2)-N(2))^2)< Costetotal(N(1),N(2))
        movimiento = true;
    else movimiento = false;
    end
else movimiento = false;
end
% Comprobamos que se cumplen las condiciones generales (no obstaculos en MAP y
no esta en NC) y las que dependen de "direccion":
if movimiento == true & isempty(find(NC(:,1)==N(1) & NC(:,2)==N(2))) &
MAP(N(1),N(2))==0
    %Actualizamos coste:
    if CosteO(N_sel(1),N_sel(2))+coste < CosteO(N(1),N(2))
        CosteO(N(1),N(2)) = CosteO(N_sel(1),N_sel(2))+coste;
    end
    %if sqrt((PF(1)-N(1))^2+(PF(2)-N(2))^2) < CosteF(N(1),N(2))
```

```
    %    CosteF(N(1),N(2)) = sqrt((PF(1)-N(1))^2+(PF(2)-N(2))^2);
    %end
    if costeheuristica(PF,N)<CosteF(N(1),N(2))
        CosteF(N(1),N(2)) = costeheuristica(PF,N);
    end
    Costetotal(N(1),N(2))=CosteO(N(1),N(2))+CosteF(N(1),N(2));
    %Actualizamos NA:
    if isempty(find(NA(:,1)==N(1)&NA(:,2)==N(2)))==1
%Buscamos si N esta en NA.
        NA=[NA;N ,CosteO(N(1),N(2)),CosteF(N(1),N(2)),Costetotal(N(1),N(2))];
%Sí no esta -> Introducimos N.
    else
        NA(find(NA(:,1)==N(1)&NA(:,2)==N(2)),3:5) =
[CosteO(N(1),N(2)),CosteF(N(1),N(2)),Costetotal(N(1),N(2))];    %Sí esta ->
Cambiamos sus costes.
    end
    %Actualizamos predecesores%
    if isempty(find(Predecesores(:,3)==N(1)&Predecesores(:,4)==N(2)))==1
%Buscamos si N esta en Predecesores como un destino.
        Predecesores=[Predecesores;N_sel(1),N_sel(2),N(1),N(2)];            %Sí
no esta -> Introducimos N_sel como origen y N como destino.
    else
        Predecesores(find(Predecesores(:,3)==N(1)&Predecesores(:,4)==N(2)),1:2)
= N_sel;    %Sí esta -> Cambiamos el origen para llegar a N_n.
    end
    %Actualizamos N_prev:%
    if N(1)==PF(1) && N(2)==PF(2)
        N_prev=N_sel;
    end
end
end
```

6.2.3. costeheuristica.m

```
function [ coste ] = costeheuristica(PF, N)
%costeheuristica: Calcula el coste de la función heurística para el algoritmo
A*.
%   PF -> Punto final.
%   N -> Punto que queremos estudiar.
Y=PF(1)-N(1);
X=PF(2)-N(2);
if Y<0
    Y=N(1)-PF(1);
end
if X<0
    X=N(2)-PF(2);
end
Punto=[Y,X];
Punto=sort(Punto);
coste=sqrt(2)*Punto(1)+1*(Punto(2)-Punto(1));
end
```

6.3. Explicación del SIMROUTE y sus programas principales

Como se ha explicado anteriormente, la optimización será llevada a cabo a través del SIMROUTE, el cuál consta de un conjunto de scripts y funciones de Matlab que ejecutados ordenadamente permiten la definición de la malla según las latitudes y longitudes deseadas, así

como la definición de un nodo destino y un nodo origen para realizar la optimización de la ruta entre esos dos puntos. A continuación, se explicarán brevemente los componentes principales y más utilizados del SIMROUTE2 y que inputs son necesarios para cada uno. A continuación, hay una pequeña descripción más detallada de los scripts que componen el programa y sus inputs y outputs.

Los principales para el escenario de la ruta corta son:

- star.m
- make_mesh.m
- make_waves.m
- simroute.m
- simroute_fix.m

star.m
Script muy simple que usa la función *addpath* con todas las carpetas necesarias que se usarán más adelante.

make_mesh.m
Crea el mallado del mar mediterráneo en la sección que seleccionemos mediante la definición de las longitudes y latitudes máxima y mínima. Además, nos permite definir la resolución del mallado, es decir, la mayor o menor cantidad de nodos que deseamos.

make_waves.m
Utiliza los datos del opendap y el archivo creado por *make_mesh.m* y crea un archivo .mat con el oleaje y la dirección de este para la malla creada anteriormente. Además, permite definir la hora de inicio del viaje desde el puerto de origen. Por tanto, como inputs es necesario definirle la hora de inicio y el nombre del archivo descargado del opendap.

simroute.m
Se trata del script principal el cual se encarga de realizar el cálculo de la ruta óptima. Los inputs necesarios para poder realizar los cálculos de optimización son los siguientes: nodo de inicio, nodo final, nombre del archivo generado por make_waves.m, velocidad de crucero del navío, indicar si utiliza el algoritmo Dijkstra o A*. En cuanto a los nodos inicio y final, para calcularlos hay que usar la función *search_nods.m*, la cual si le damos la latitud y longitud de un punto del mapa nos devuelve el nodo que representa en el mallado. El mismo *simroute.m* se encarga de avisarnos si el nodo seleccionado se trata de tierra firme o mar.

simroute_fix.m
Hace los cálculos para el caso de la ruta de distancia mínima usando los archivos generados por el *simroute.m* y el *make_waves.m*.

Además, existen 4 scripts más que son muy útiles a la hora de realizar los cálculos para el escenario de la ruta larga, en la que se trabaja con un mallado de todo el mediterráneo en lugar de una sección de este, estos scripts son:

- make_mesh_MED
- make_waves_MED
- make_sea_step1
- make_sea_step2

make_mesh_MED.m
Funciona igual que el *make_mesh.m* pero directamente ya tiene puestas las latitudes y longitudes máximas y mínimas que delimitan el mediterráneo. Se puede seguir definiendo la resolución como en el caso del *make_mesh.m*.

make_waves_MED.m
Funciona igual que el make_waves.m pero en este caso no es posible cambiar la hora de inicio del viaje que está fijada a las 00:00h.

make_sea_step1.m
Sirve, junto a *make_sea_step2.m,* para abrir los estrechos o zonas en que debido a la interpolación que hace el *make_waves.m* no contemplan agua en lugares en que debería haber. *make_sea_setp1.m* nos permite seleccionar la zona del mediterráneo en que deseamos llevar a cabo la operación.

make_sea_step2.m
Después de ejecutar el *make_sea_step1.m* el *make_sea_step2.m* nos permite, en la zona designada en el primer paso, definir el rectángulo específico en el que se desea hacer una nueva interpolación más detallada para conocer los datos del oleaje en esos puntos.
Podemos diferenciar el *make_sea_step2_ew.m* y el *make_sea_step2_ns.m* en función de si el rectángulo a interpolar realiza la interpolación de este a oeste o de norte a sur, es decir, para realizar la obertura de un canal horizontal o vertical, respectivamente.

Además de los scripts, existen dos funciones muy útiles a la hora de visualizar los resultados obtenidos:

- plot_routes
- plot_routes_amd_waves

plot_routes.m
Dándole los dos archivos obtenidos después de aplicar el *simroute.m* y el *simroute_fix.m* realiza la representación de las dos rutas en la zona del mediterráneo definida previamente en *make_mesh.m*. Únicamente representa el contorno de la costa y las rutas.

plot_routes_amd_waves.m
Esta segunda función va representando y guardando la evolución del oleaje, así como de la ruta al paso de cada hora hasta llegar al nodo de destino.